BMR 麦克泰技术 芯来科技 NUCLEI GigaDevice

“国产MCU开发技术与生态建设”

第三期嵌入式与物联网开发技术

线上分享课程

第5讲：开源软件助力国产MCU -GD32上体验RTOS和IOT OS

主讲人：何小庆/Allan He

2020年/8月/20日

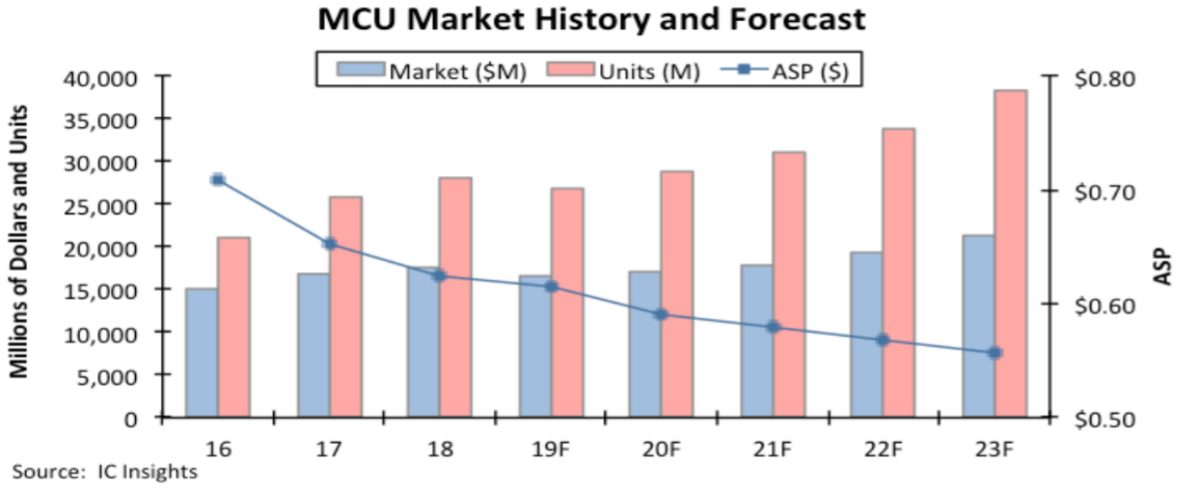
单片机与嵌入式系统应用 Microcontrollers & Embedded Systems
eefocus 与非网

演讲内容

- 中国MCU市场与国产MCU发展趋势
- 开源软件：RTOS 与物联网操作系统
- GD32VF103上体验FreeRTOS
- GD32VF103上体验embOS和emWin
- GD32450上体验 Amazon FreeRTOS

全球MCU 市场分析

- 在2019年下滑之后，预计2020年MCU市场将出现温和反弹，增长3.2%，达到约**171亿**美元，而出货量预计将增长7%以上创下了**289亿**颗的新高记录
- 2018年至2023年的复合年增长率（CAGR）为3.9%，2023年将达到**213亿**美元
- 预计未来五年，MCU单位出货量的复合年增长率为6.3%，达到**382亿**颗
- 更多传感器和设备接入 IoT催生收入和数量增长，但32位MCU 激烈竞争，**ASP**大幅降低



国产 MCU 的 现状

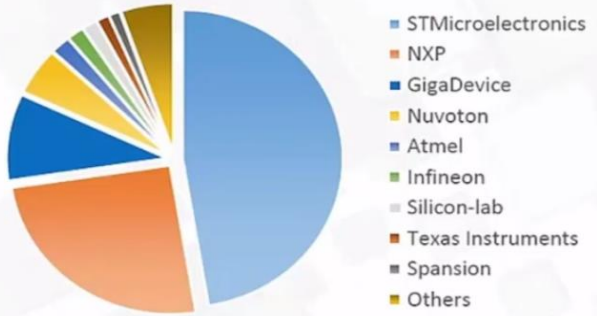
- 中国市场OEM厂商需求大，ARM内核授权普及，政策刺激，RISC-V 兴起，催生了MCU初创企业
- **生态与核心技术**是多数国产MCU弱项，多数国产MCU停留在开发板、烧写器和基础固件库老三样上。IDE、RTOS和中间件依靠第三方的支撑。**传感、模拟、安全、无线、专业算法库**等核心技术与国外MCU大厂依旧有较大差距

部分国产通用MCU厂商

国产 MCU 发展机遇

- 32位MCU 需求持续成长，迎来了前所未有的机遇
  - **AIOT 时代**, 8/16 位MCU 的算能已达瓶颈, 32位MCU 的**价格及功耗**日渐变低带动需求
- 国产MCU走向前台并在国际市场有一席之地，将是一条艰难曲折之路

China MCU Business and Ranking



2018 China Cortex MCU Market Share Estimation (Based on Bill to China)
Source: IHS Markit , Mar. 2019

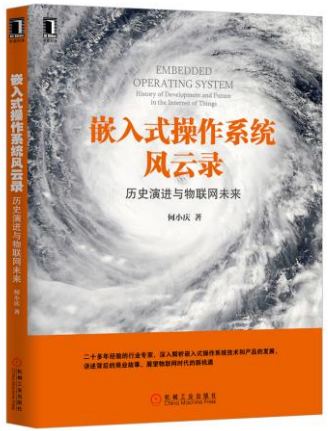
2018 China Cortex-M Market Share Estimation		
1	STMicroelectronics	47.4%
2	NXP	25.2%
3	GigaDevice	9.4%
4	Nuvoton	5.1%
5	Atmel	2.0%
6	Infineon	1.7%
7	Silicon-lab	1.5%
8	Texas Instruments	1.3%
9	Spansion	1.1%
10	Others	5.3%

MCU开发在AIOT时代的变化

处理器和IP	软件和工具	生态系统
• **更高的处理能力** • **更多的安全组件** • 多种连接能力 • 更低功耗	• **操作系统从任务调度发展为IoT OS平台** • 软件复杂度大幅增加 • 平台级软件及工具	• **各种云服务公司进入嵌入式系统生态圈** • 与算法公司，纯软件公司合作增多

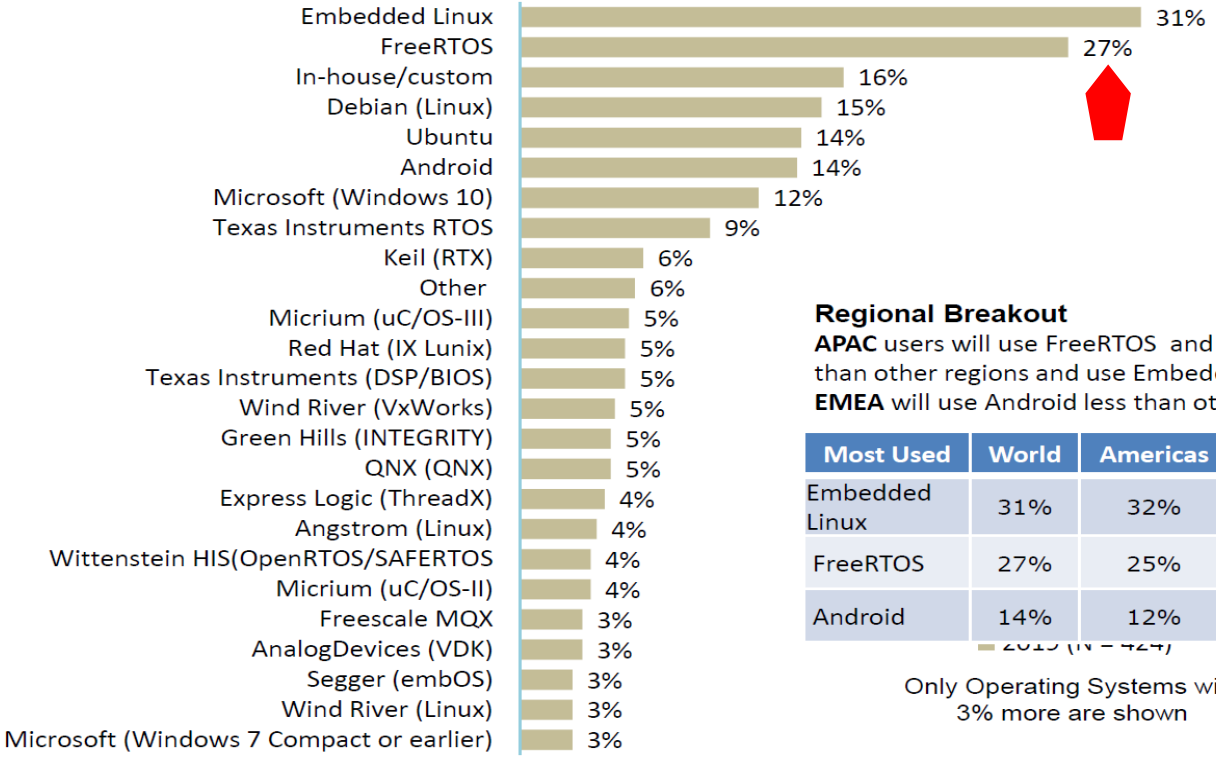
开源的RTOS

- RTEMS
  - 实时多处理器系统，最早运用在美国国防系统
- TOPPERS
  - 日本著名开源的RTOS，创始人是高田教授，专注在汽车电子
- **μC/OS**
  - 最新开源的老牌商业RTOS, 图书资料丰富，国内高校印象力大
- **FreeRTOS**
  - 其目标在支持MCU ， 开源模式和生态好，配套资料少
- Nutt
  - API完全符合POSIX标准,实时性和配套丰富，在无人机有名气
- **Zephyr**
  - Linux基金会的一个微内核项目,由Intel 主导，面向IoT安全
- **RT-Thread**
  - 中国自己开发的开源RTOS ， 国内知名度很高



RTOS 有超过30年历史，开源RTOS 发挥重要作用

2019 嵌入式 OS 市场调查



Regional Breakout

APAC users will use FreeRTOS and Android much more than other regions and use Embedded Linux much less.
EMEA will use Android less than other regions.

Most Used	World	Americas	EMEA	APAC
Embedded Linux	31%	32%	31%	26%
FreeRTOS	27%	25%	24%	37%
Android	14%	12%	10%	26%

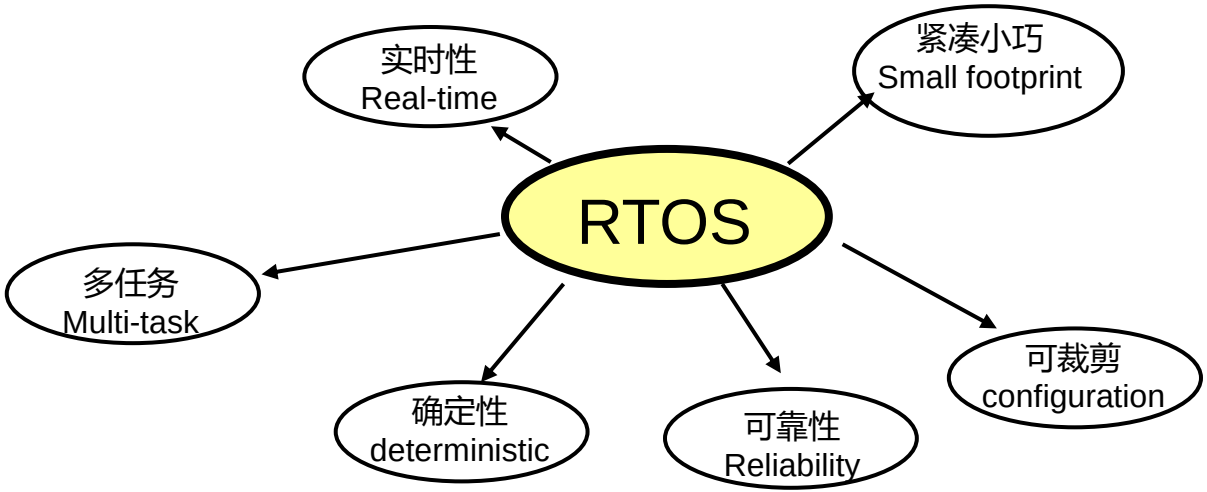
Only Operating Systems with 3% more are shown

来自：EETimes embedded

未来12个月你可能使用的Embedded OS

什么是RTOS?

R(real) T(time) OS 实时多任务操作系统、RTOS是一种操作系统，属于嵌入式操作系统，RTOS种类很多:有商业的、DIY和开源的

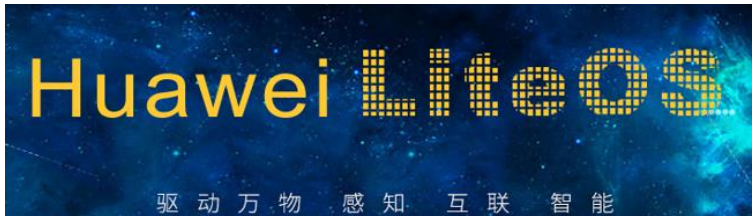


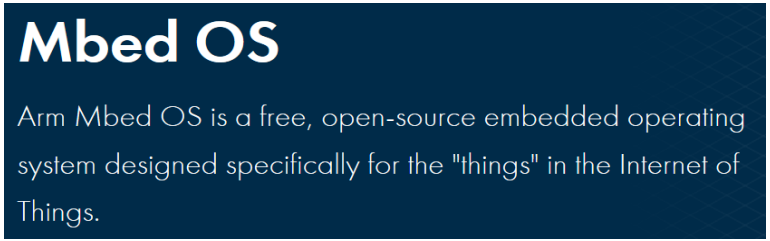
RTOS 带来什么好处?

- RTOS 为你的应用提供**服务（API）**
  - 任务、队列、存储和时间管理
- RTOS方便增加**软件组件**
  - 为应用带来更多的便利- IoT 和HMI
- RTOS 是你开发应用的一个**基础架构**
  - 支持不同CPU 和 MCU
- 无论是商业还是开源RTOS bug 很少
  - 许多RTOS 通过**安全认证**
  - DO-178B, IEC-61508, IEC-62304
- RTOS 很容易支持电源管理机制
  - 许多RTOS 内核有**低功耗调度**功能

什么是IOT OS?

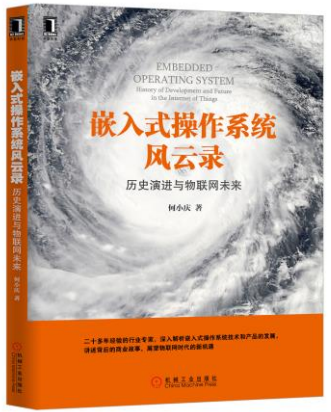
- **物联网操作系统** 国外称为The Operating System for Internet of Things，简称为**IoT OS**。它是一种在嵌入式实时操作系统（**RTOS**)基础上发展出来的、面向IoT架构和应用场景的 **软件平台**
- 物联网操作系统目前没有严格的定义，体系架构和功能各有不同，种类也比较多；有运行在 MCU, 比如 Amazon FreeRTOS，有运行在嵌入式处理器（MPU上，比如Android things，还有是传统的RTOS改进后的，比如RT-Thread





IOT OS 的技术特征

- **管理物的能力**
  - “物” 是“嵌入式实时的低功耗传感器设备”
- **泛在的通信能力**
  - 支持各种无线和有线，近场和远距离的通信协议
- **设备的可管理和维护性**
  - 支持设备的安全动态升级和远程维护
- **物联网安全**
  - 物联网安全包含设备、通信和云安全，具备防御外部安全入侵和篡改能力
- **物联网云平台**
  - 通过云物联网平台完成远程设备管理，数据存储和分析，安全控制和业务支撑，这是物联网大数据和人工智能的基础

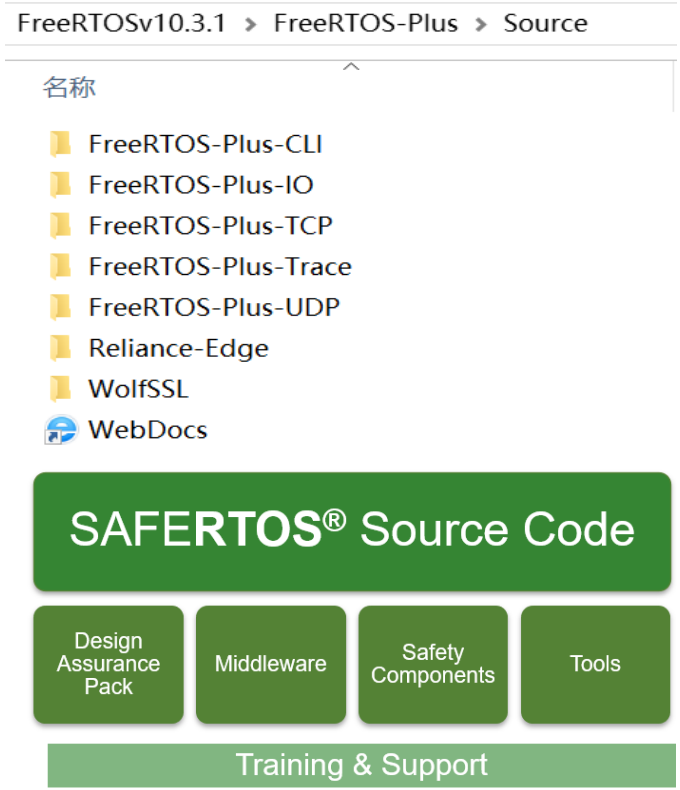


FreeRTOS介绍

- 英国人**Richard Barry 2003**年发布的开源的实时内核-FreeRTOS
- FreeRTOS 支持超过35 CPU 架构，**每3 分钟就有一次下载**，FreeRTOS成为世界最受开发者欢迎的RTOS
- IoT把FreeRTOS推到了风口浪尖，各家MCU芯片公司都在了FreeRTOS
- **2017年**FreeRTOS 成为亚马逊 **AWS 开源项目**，Richard 成为亚马逊 AWS IoT 首席架构师。FreeRTOS v10.0 采用 MIT 授权方式 ,由亚马逊托管， 亚马逊发布了 Amzon FreeRTOS 1.0
- Amazon FreeRTOS 使用FreeRTOS v10.0 内核，增加了IoT应用组件，比如OTA、TLS 和 MQTT 方便连接到**亚马逊AWS云**上
- 系列产品：**FreeRTOS**、**OpenRTOS**、**SaftRTOS** 和 **Amazon FreeRTOS**
- 最新版本是 Amzon Version 202002.00 FreeRTOS 10.3

FreeRTOS软件生态系统

- **FreeRTOS Plus** - FreeRTOS 中间件
  - FreeRTOS+CLI
    - 可扩展的命令行接口framework
  - FreeRTOS+TRACE
  - FreeRTOS+TCP& FreeRTOS+UDP
  - FreeRTOS+IO
    - 在外设驱动及应用层之间提供POSIX接口层
- RelianceEdge 开源安全文件 和WolfSSL 开源安全协议
- **OPENRTOS**
  - OPENRTOS是FreeRTOS的商业授权版本，代码与FreeRTOS完全一致
- **SAFERTOS**
  - 基于FreeRTOS功能模型的RTOS内核
  - 通过IEC61508 SIL3&ISO 26262 ASIL D预认证

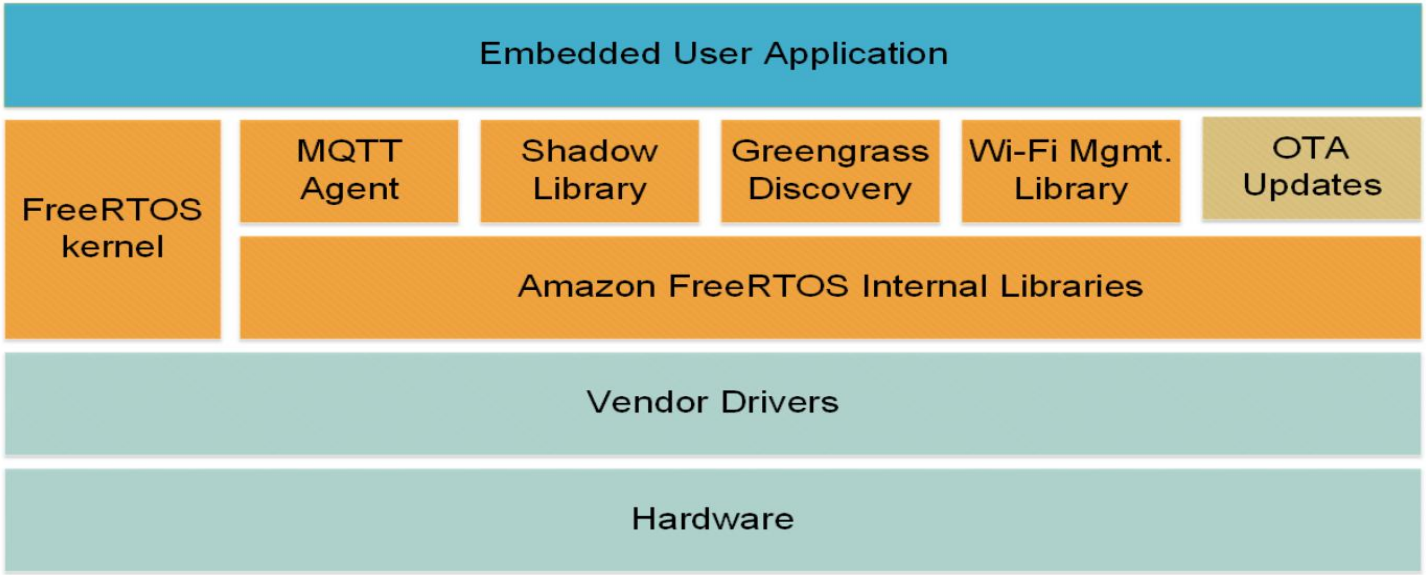


FreeRTOS 技术特征

- 架构简洁，支持超过30多种处理器架构
- 支持可抢占式/协作式/时间片任务调度
- 通信和同步机制包含任务通知、队列(信号量/互斥量/递归式互斥量)、事件标志
- 互斥量支持优先级继承
- 支持软件定时器
- 支持tTickless模式（低功耗管理）
- 执行跟踪功能和堆栈溢出检测
- 紧凑的尺寸（4-9KB）适应8-16-32位MCU

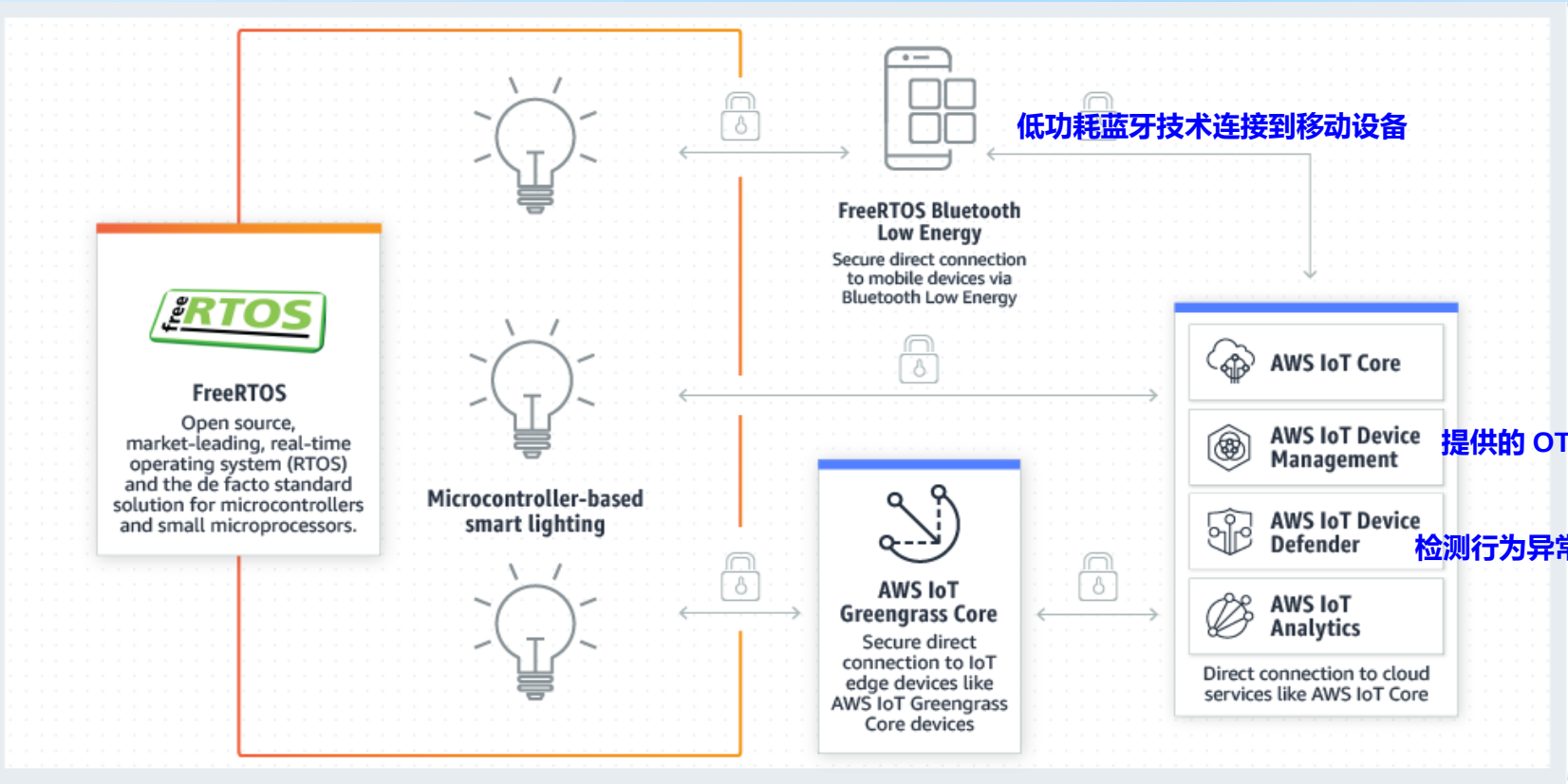
Amazon FreeRTOS

- Amazon FreeRTOS 是一款适用于微控制器的操作系统，可让您轻松地对低功耗的小型边缘设备进行编程、部署、安全保护、连接和管理。Amazon FreeRTOS 以 **FreeRTOS 内核**为基础，并通过**软件库对其进行扩展**，从而让您可以轻松地将小型低功耗设备**安全连接到 AWS 云服务**或运行 **AWS Greengrass** 的功能更强大的边缘设备，MIT授权开源软件



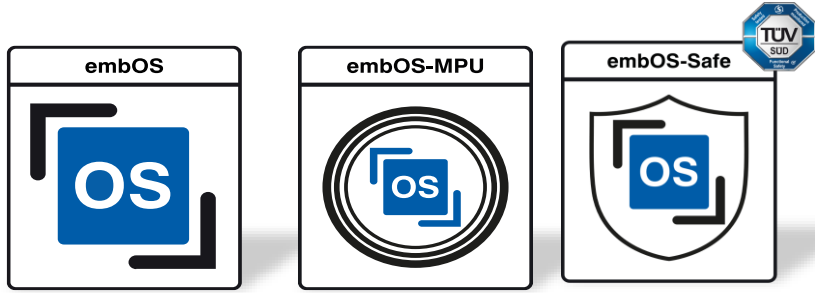
https://github.com/aws/amazon-freertos

Amazon FreeRTOS 工作原理



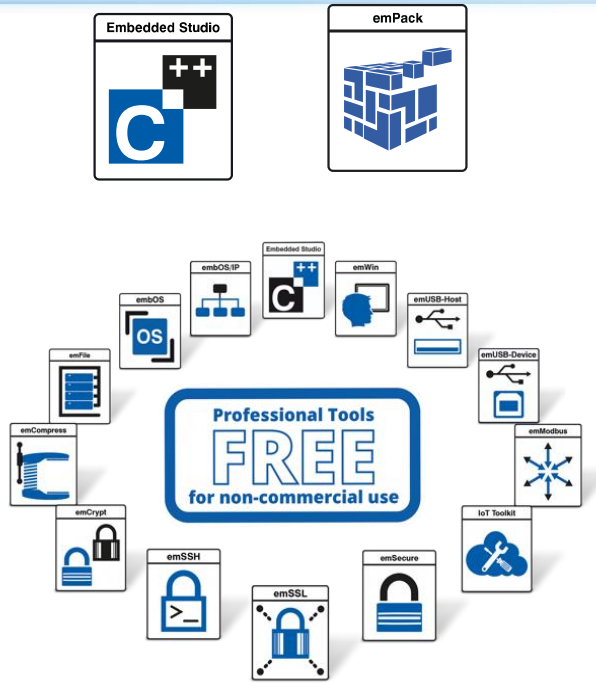
embOS –全球知名的RTOS

- 超过25年的持续发展，数十亿个设备中进行了验证
- 100%的代码由SEGGER（德国制造）编写
- 最高开发标准：
  - -符合MISRA-C：2012标准
  - -通过IEC 61508 SIL3和IEC 62304 C类认证
- 最小的RAM和ROM使用
  - 内核ROM大小：1700字节，
  - 内核RAM大小：67字节
  - 每个任务控制块的RAM使用量：36个字节
- 广泛的低功耗支持 如Tickless
- 当前80个embOS移植和500个板级支持软件包



Embedded Studio PRO

- 高性能RTOS embOS
- 强大的文件系统emFile
- 高效的图形软件包**emWin**
- 多功能USB堆栈emUSB设备和emUSB主机
- 高性能IP协议 embOS / IP
- 安全软件emSecure，emCrypt，emSSL，emSSH
- 物联网工具包
- 压缩emCompress
- 完整的开发环境**Embedded Studio**



All-in-one package

RISC-V® Architecture

Embedded

DEMO ON GD32VF103

GD32 RISC-V开发板

- **GD32VF103-EVAL 开发板**
  - 主控 GD32VF103VBT6，基于芯来科技Bumblebee RISC-V 处理器（RV32IMAC），主频108MHz
  - **128KB Flash，32KB SRAM**
  - 3.2寸 TFT LCD，240x320分辨率，触摸屏
  - 外设：
    - Timer（高级16位定时器，通用16位定时器）、UART、I2C、SPI/I2S、CAN、USBFS、ADC/DAC、EXMC、GPIO
  - SPI Flash （GD25Q16）
  - 板载 GDLink调试器，标准20 线 JTAG接口，支持**J-Link调试器**，调试速度要更快
- 适用于工业控制、消费电子、新兴IoT嵌入式市场应用

Educational Unit

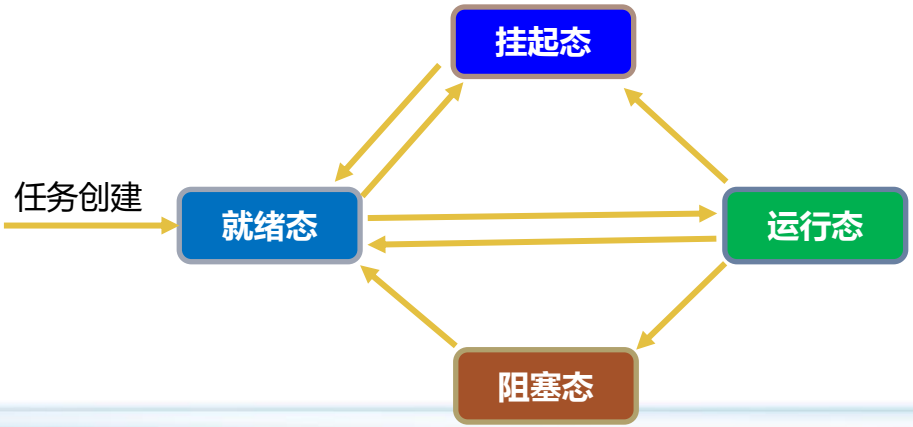
任务

- 什么是任务?
  - 在 FreeRTOS 中，每个执行线程都被称为"任务"。在嵌入式社区中，对此并没有一个公允的术语
  - 每个任务都是在自己权限范围内的一个小程序。其具有程序入口，通常会运行在一个死循环中，也不能退出
  - 每个任务拥有属于自己的栈空间，以及属于自己的自动变量(栈变量)
  - 每个创建的任务需要分配一个任务控制块(TCB)，用于保存到任务的所有信息，如堆栈指针、任务名称等
- 创建任务
  - 将任务堆栈、任务控制块和任务函数建立联系，RTOS内核对任务进行管理和调度，使用任务创建API创建任务

22

任务的状态

- FreeRTOS任务的状态
  - FreeRTOS 系统中的每一个任务都有四种运行状态
  - 任务被创建后处于就绪状态，就绪的任务已经具备执行的条件，只等待调度器进行调度
  - 发生任务切换时，就绪态任务中优先级最高的任务被执行，进入运行态



23

任务间通信

- 任务间的通信
  - 任务间通信机制包括**信号量、消息队列、互斥量、递归信号量、事件标志，通知**
  - FreeRTOS 中所有的通信与同步机制都是基于队列实现的
  - 信号量用于任务与任务，任务与中断的同步，包含共享资源
  - 互斥量用于任务之间需要互斥的共享资源的保护
- 消息队列
  - 队列可以保存有限个具有确定长度的数据单元，可以保存的最大单元数目被称为队列的“深度”
  - 通常情况下队列被作为 FIFO(先进先出)使用，但也可以用作LIFO(后进先出)
  - 往队列写入数据是通过字节拷贝把数据复制存储到队列中；从队列读出数据使得把队列中的数据拷贝删除

24

创建任务

- 创建任务的方法
  - 创建任务使用 FreeRTOS 的 API 函数 xTaskCreate()

```
BaseType_t xTaskCreate( TaskFunction_t    pvTaskCode,     /* 任务函数          */
                        const char *const pcName,         /* 任务名            */
                        unsigned short    usStackDepth,   /* 任务栈大小，单位是字*/
                        void *            pvParameters,   /* 任务参数          */
                        UBaseType_t       uxPriority,     /* 任务优先级        */
                        TaskHandle_t *    pvCreatedTask );/* 任务句柄          */
```

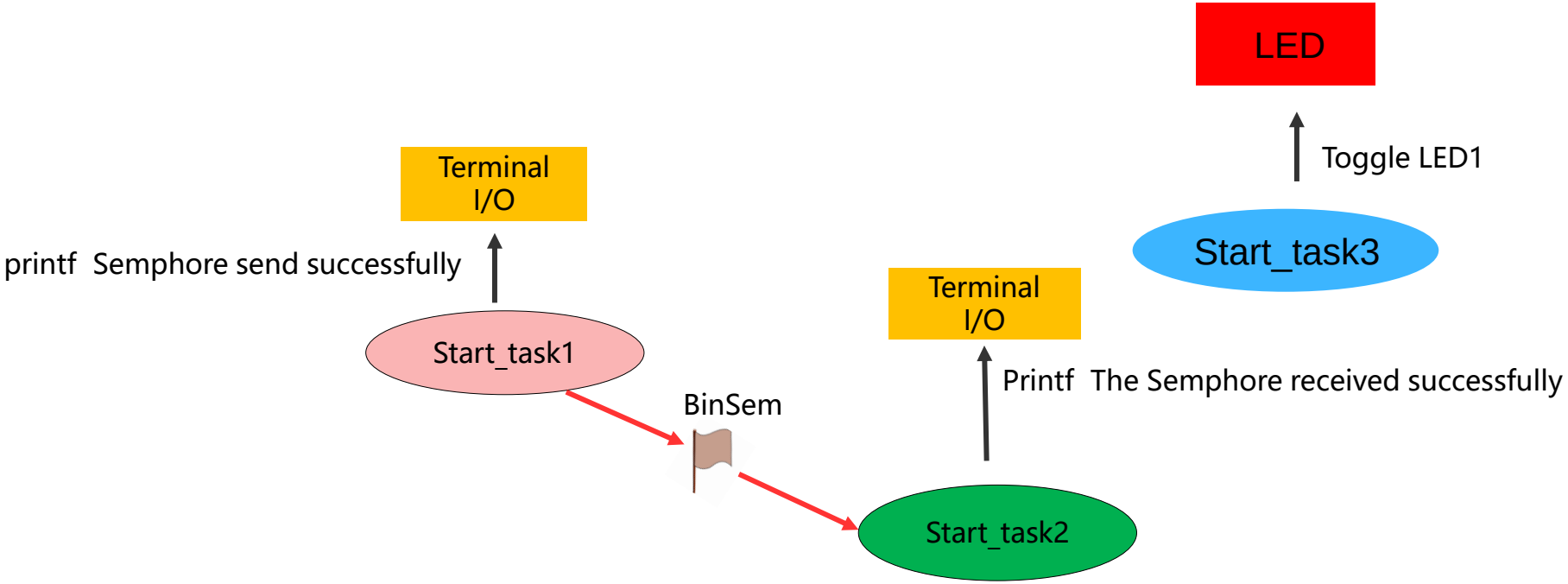
- FreeRTOS的优先级数值越大，其逻辑优先级越高
- 任务名只是方便开发人员调试时查看
- 任务句柄用于区分不同的任务，其实质是任务控制块的地址

25

创建二值信号量

- 创建二值信号量
  - 信号量有二值信号量和计数信号量，二值信号量可以看作是一个深度为 1 的队列。这个队列由于最多只能保存一个数据单元，所以其不为空则为满(所谓"二值")
  - **计数信号量可以看作是深度大于 1 的队列，典型的用法是用作事件计数和资源管理**
  - 调用API函数**SemaphoreHandle_t xSemaphoreCreateBinary(void)**创建二值信号量
  - 如果创建成功会返回二值信号量的句柄，如果由于 FreeRTOSConfig.h 文件中 heap 大小不足，无法为此二值信号量提供所需的空间会返回 NULL

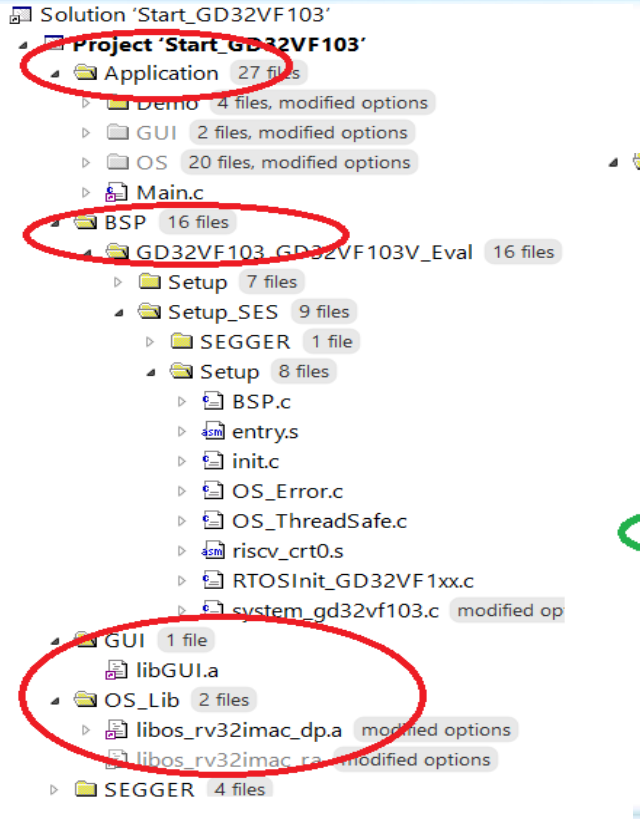
实验原理框图

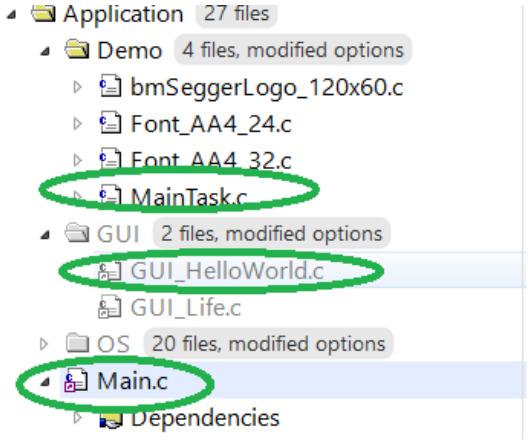


更加详细原理和移植介绍 参见第二期嵌入式与物联网开发技术线上课程 （网易云课堂）

embOS 和 emWin demo （1）

- DEMO 构建在SES 工程里面
- 包含：
  - OS lib 库
  - GUI Lib库
  - 初始化和BSP
  - 应用（DEMO) 代码





embOS 和 emWin demo （2）

- demo 建立了2个应用任务一个 GUI 任务
- HP和LP 是2个优先级相同的定时任务

```
int main(void) {
  OS_InitKern();                      /* Initialize OS                    */
  OS_InitHW();                        /* Initialize Hardware for OS       */
  BSP_Init();                         /* Initialize LED ports             */
  BSP_SetLED(0);                      /* Initially set LED                */
  /* You need to create at least one task before calling OS_Start() */
  OS_CREATETASK(&TCB0, "MainTask", MainTask, 100, Stack0);
  OS_CREATETASK(&TCBHP, "HP Task", HPTask, 50, StackHP);
  OS_CREATETASK(&TCBLP, "LP Task", LPTask,  50, StackLP);
  OS_Start();                         /* Start multitasking               */
  return 0;
}
```

```
LP task ok
HP task ok
HP task ok
HP task ok
HP task ok
LP task ok
HP task ok
HP task ok
```

```
*       MainTask
*/
void MainTask(void) {
  GUI_Init();
  WM_CreateWindowAsChild(0, 0, LCD_GetXSize(), LCD_GetYSize
  while (1) {
    GUI_Delay(100);
  }
}
```

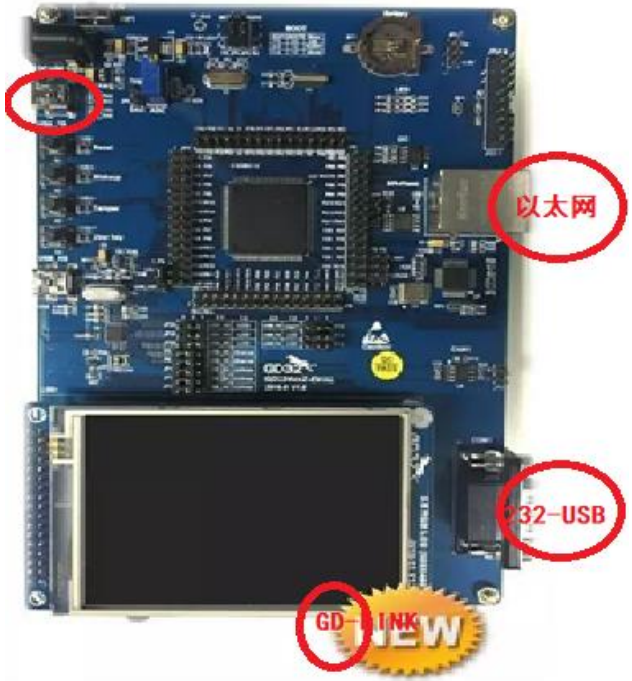
Cortex™
Intelligent Processors by ARM®

Embedded

DEMO ON GD32450X

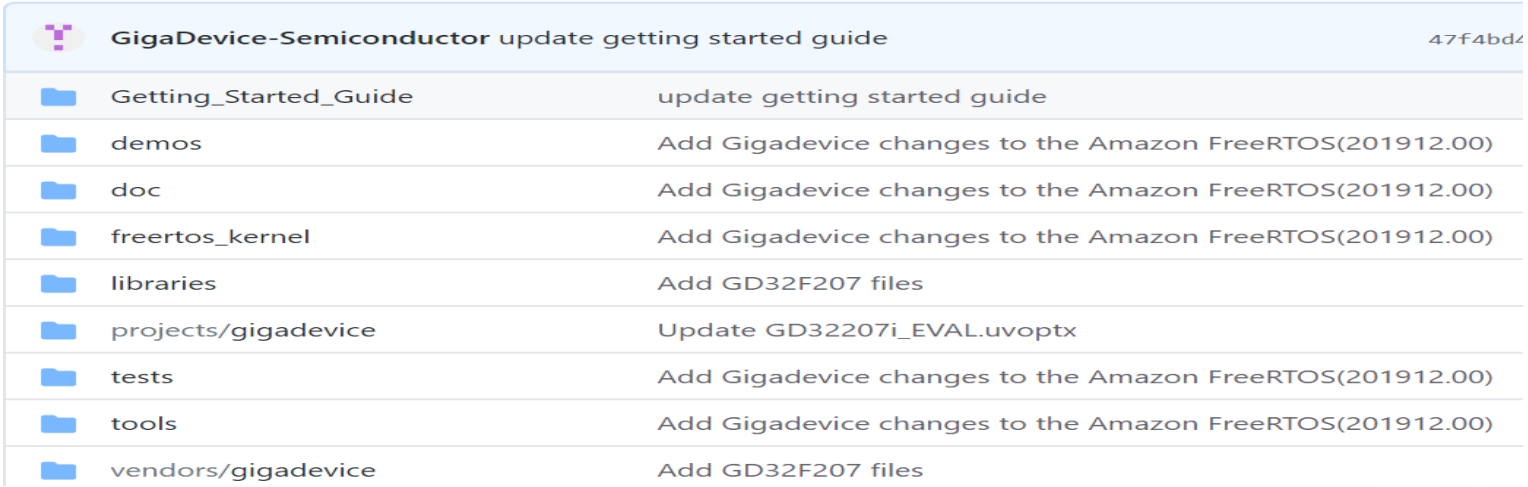
GD32 ARM 开发板

- **GD32450Z-EVAL 开发板**
  - 主控芯片GD32450ZK ARM Cortex M4 MCU
    200 MHZ
    512K FLASH 256K SRAM
  - 外设：
    - Timer，GPIO，UART，SPI/I2C，
      ENET,CAN,USB FS+H,ADC, DAC和 EXMC
  - 扩展了 4.3 寸 480x270 的TFT LCD 触摸屏
  - SPI FLASH (GD25Q40)
  - 板载 GDLink，20线标准 JTAG 调试接口
- 适用于工业控制、电机变频、图形显示、安防监控、传感器网络、无人机、机器人、物联网等高性能计算应用场合
  -



GD32 MCU Amazon FreeRTOS 实例

- 安装 KEIL IDE 并配置开发环境
  - GD32F4xx 系列芯片的 Pack 包，通过访问http://www.gd32mcu.com/cn/download/7
- 编译从 GitHub 上下载的源码
  - https://github.com/GigaDevice-Semiconductor/amazon-freertos
- 配置，修改参数，运行示例代码

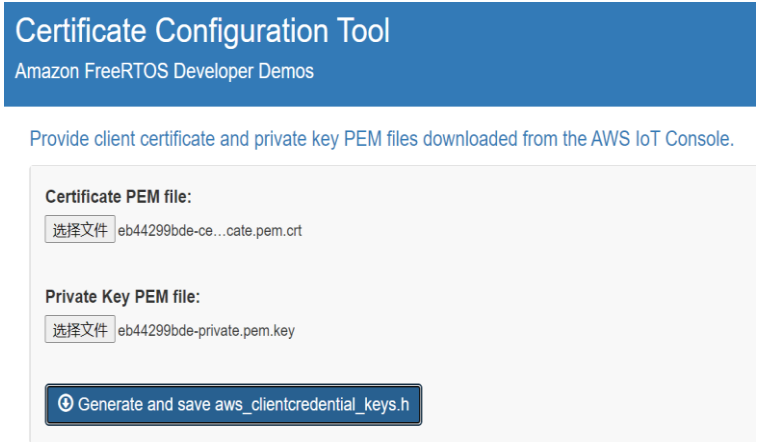


AWS 云端侧开发

- 在AWS IOT 上注册你的GD32450Z-EVAL设备
  - 注册AWS IAM 用户
    - AWS中国提供服务
  - 建立一个IOT THINGS（物）
    - AWS IOT 可以管理你的设备
  - 建立一个私钥和证书（certificates）
    - 让你的设备可以得到AWS IOT认证
  - 建立一个AWS IOT POLICY（策略）
    - 让你的设备获得访问AWS IOT 资源服务的授权
  - 将-物-证书-策略关联起来

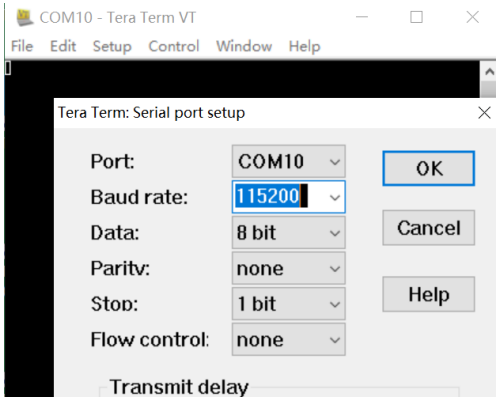
AWS 开发板端侧开发

- 你需要将私钥和证书经 AWS IOT 工具 CertificateConfigurator.html转换存入 AWS 客户凭证文件中
- 存入 Amazon freeRTOS\amazone-freertos\demos\include 的 目录
- 替代原来的 aws_clientcredential.h
- 修改部分配置代码
  - *Freertosipconfig.h 141 行 #defineipconfigUSE_DHCP 1*
  - *aws_clientcredential.h 34 行 填写 AWS端点地址 41 行 填写物名*
  - *可能存在RAM 空间不足的问题*
  - *FreeRTOSconfig.h*
  - *88 行 configtotal_heap_size 140*
  - *191 行 ipconfigNUM_NETWORK_BUFFER_DESCRIPTORS 10*
  - *在FreeRTOS_Plus_TCP目录下Bufferallocation_2.c 替换 Bufferallocation_1.c 也可牺牲效率换取RAM利用率*
- KEIL IDE 构建-下载-运行




```
34  #define clientcredentialMQTT_BROKER_ENDPOINT                    s.iot.cn
35
36  /*
37   * @brief Host name.
38   *
39   * @todo Set this to the unique name of your IoT Thing.
40   */
41  #define clientcredentialIOT_THING_NAME                "GD32450Z"
```

运行示例代码的结果- 开发板

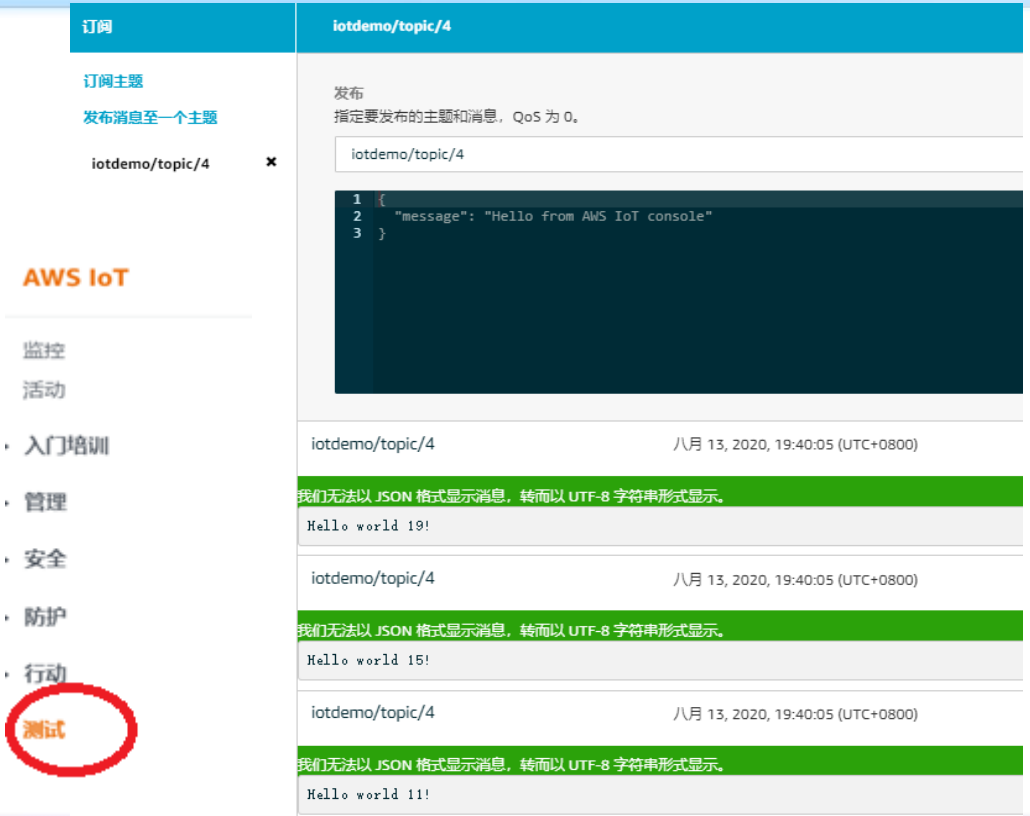



```
Subscription topic filter: iotdemo/topic/1
Publish topic name: iotdemo/topic/1
Publish retain flag: 0
Publish QoS: 1
Publish payload: Hello world 0!
[INFO ][MQTT][4814] (MQTT connection 20015ca0) MQTT PUBLISH operation queued.
[INFO ][DEMO][4821] Acknowledgment message for PUBLISH 0 will be sent.
[INFO ][DEMO][4827] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/2
Publish topic name: iotdemo/topic/2
Publish retain flag: 0
Publish QoS: 1
Publish payload: Hello world 1!
```

```
FreeRTOS App Ver:90002
FreeRTOS_IPInit
vTaskStartScheduler
Network buffers: 10 lowest 10
Network buffers: 9 lowest 9
vDHCPProcess: offer c0a81f2eip
vDHCPProcess: offer c0a81f2eip
Write certificate...

IP Address: 192.168.31.46
Subnet Mask: 255.255.255.0
Gateway Address: 192.168.31.1
DNS Server Address: 192.168.31.1

[INFO ][DEMO][2185] ---------STARTING DEMO---------

[INFO ][INIT][2190] SDK successfully initialized.
[INFO ][DEMO][2194] Successfully initialized the demo. Network type for the demo: 4
[INFO ][MQTT][2202] MQTT library successfully initialized.
[INFO ][DEMO][2207] MQTT demo client identifier is 00234507 (length 8).
DNS[0xC96E]: The answer to 'ak42ahou55ic7-ats.iot.cn-northwest-1.amazonaws.com.cn' (24535a21ip) will be stored
Network buffers: 8 lowest 8
[INFO ][MQTT][4596] Establishing new MQTT connection.
[INFO ][MQTT][4600] Anonymous metrics (SDK language, SDK version) will be provided to AWS IoT. Recompile with AWS
[INFO ][MQTT][4614] (MQTT connection 20015ca0, CONNECT operation 20015e20) Waiting for operation completion.
[INFO ][MQTT][4691] (MQTT connection 20015ca0, CONNECT operation 20015e20) Wait complete with result SUCCESS.
[INFO ][MQTT][4701] New MQTT connection 2000d7e8 established.
[INFO ][MQTT][4706] (MQTT connection 20015ca0) SUBSCRIBE operation scheduled.
[INFO ][MQTT][4713] (MQTT connection 20015ca0, SUBSCRIBE operation 20015e40) Waiting for operation completion.
[INFO ][MQTT][4771] (MQTT connection 20015ca0, SUBSCRIBE operation 20015e40) Wait complete with result SUCCESS.
[INFO ][DEMO][4781] All demo topic filter subscriptions accepted.
[INFO ][DEMO][4787] Publishing messages 0 to 1.
[INFO ][MQTT][4791] (MQTT connection 20015ca0) MQTT PUBLISH operation queued.
[INFO ][MQTT][4798] (MQTT connection 20015ca0) MQTT PUBLISH operation queued.
[INFO ][DEMO][4805] Waiting for 2 publishes to be received.
[INFO ][DEMO][4827] MQTT PUBLISH 0 successfully sent.
[INFO ][DEMO][4839] MQTT PUBLISH 1 successfully sent.
[INFO ][DEMO][4853] Incoming PUBLISH received.
```

运行示例代码的结果- 云端

- 登录 AWS IoT 控制台；
- 在控制台的导航窗口选择“测试 ”进入 MQTT 客户端；
- 在 MQTT 客户端界面中的“订阅主题”对话框中，输入对应的“iotdemo/#”，点击“订阅主题”订阅消息



Thank you !

@何小庆微博

www.hexiaoqing.net

（演讲、书、课件,和文章）

实验代码

https://eyun.baidu.com/s/3pNnvEv5

密码：x690

网易云课堂

何小庆老师的嵌入式课程