ALIBABA CLOUD

阿里云

IoT物联网操作系统
HaaS物联网通用硬件

文档版本：20220224

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.HaaS EDU K1说明书

HaaS EDU K1是HaaS Education Kit1的缩写，是基于四核高性能HaaS1000芯片打造的、集颜值和内涵于一身的物联网教育开发板。作为云端钉一体全链路解决方案的软硬件积木平台，深度集成了AliOS Things物联网操作系统、HaaS轻应用、小程序和阿里云物联网平台等技术和服务，让开发者可以轻松的学习和开发云端钉全链路实战项目，解决实际场景或孵化创新应用，图1是其全景图：

图1 HaaS EDU K1全景图

HaaS EDU K1主要特点：

1、高颜值 - 有别于传统的裸板开发板，HaaS EDU K1外观鲜艳靓丽，洋溢着青春活力。

2、可移动 - 内置1200mAh锂电池，支持移动场景，OLED屏幕和游戏键盘设计提高可玩性。

3、配置丰富 - 采用定制的四核(Cortex-A7双核1GHz和Cortex-M33双核300MHz)高性能HaaS1000芯片，自带16MB FLASH、16MB PSRAM和2.5MB SRAM，内置双频Wi-Fi和蓝牙天线，板载丰富的物联网传感器（加速度、陀螺仪、磁力计、温湿度、大气压、环境光和声音等），可开发丰富的AIoT应用场景或解决方案。

4、方便灵活 - 各接口有明确的标注，操作顺手，仅一条Type-C数据线即可完成烧录、调试和充电，非常方便。

5、可扩展 - 开发板的资源都可以灵活配置，30PIN扩展接口和SD卡槽满足更多应用场景需求

一、包装清单

图2 包装清单示意图

HaaS EDU K1的集成度比较高，如Wi-Fi和蓝牙的天线是内置的，一条Type-C数据线就可以完成充电、烧录和调试等操作，包装相对简单，具体清单如下表：

名称	个数	描述
HaaS EDU K1开发板	1	核心芯片采用HaaS1000
数据线	1	Type C
说明书	1	电子版（扫二维码）
外纸箱	1	铜版纸
内托盘	1	EVA

二、硬件规格

1、整机接口

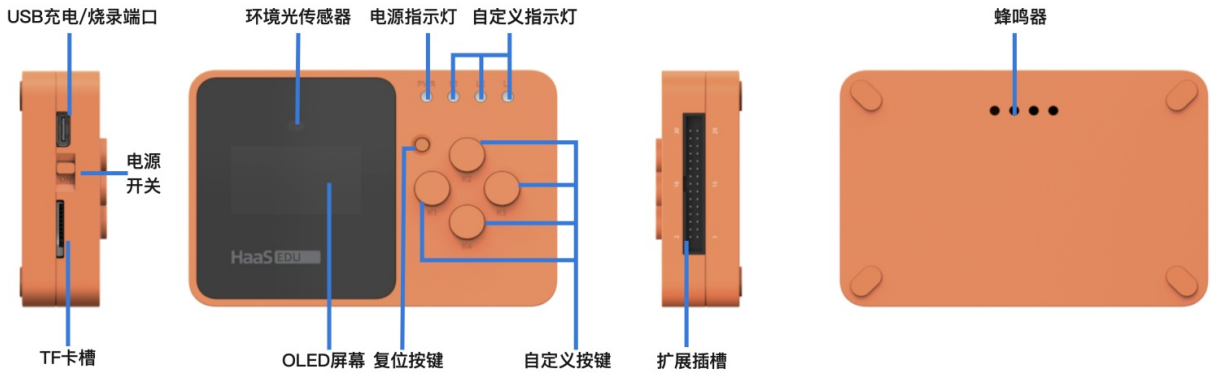


图3 整机接口示意图

如图3所示HaaS EDU K1接口资源丰富，外形大小合理：为94.4mm * 63mm * 20 mm，充分考虑手持携带的便利性。其接口定义如下表：

名称	数量	描述
OLED屏幕	1	1.3寸，分辨率128*64
环境光传感器	1	型号：AP3216C
指示灯	4	白色电源指示灯 3个可编程RGB单色灯
按键	5	1个小孔径复位按键 4个可编程按键
电源开关	1	电源ON/OFF拨动开关
TF卡槽	1	最大支持64GB
USB接口	1	Type C接口，可充电/烧录/调试
扩展卡槽	1	30PIN简牛母座
蜂鸣器排孔	4	蜂鸣器及温湿度检测对流孔

2、扩展接口

除已有板载功能之外，还有30PIN扩展接口，尽可能释放HaaS1000的资源，满足开发者更多应用需求。主要扩展有1路ADC输入、1路SWD调试、2路SPK输出、3路MIC输入、8路GPIOs等，其中GPIO_P02和GPIO_P03与主板的传感器一起复用为I2C模式，其他6个GPIOs可随便定义，并且每路GPIO的最大驱动电流是10mA，具体接口定义如图4：

图4 扩展接口示意图

3、主板功能

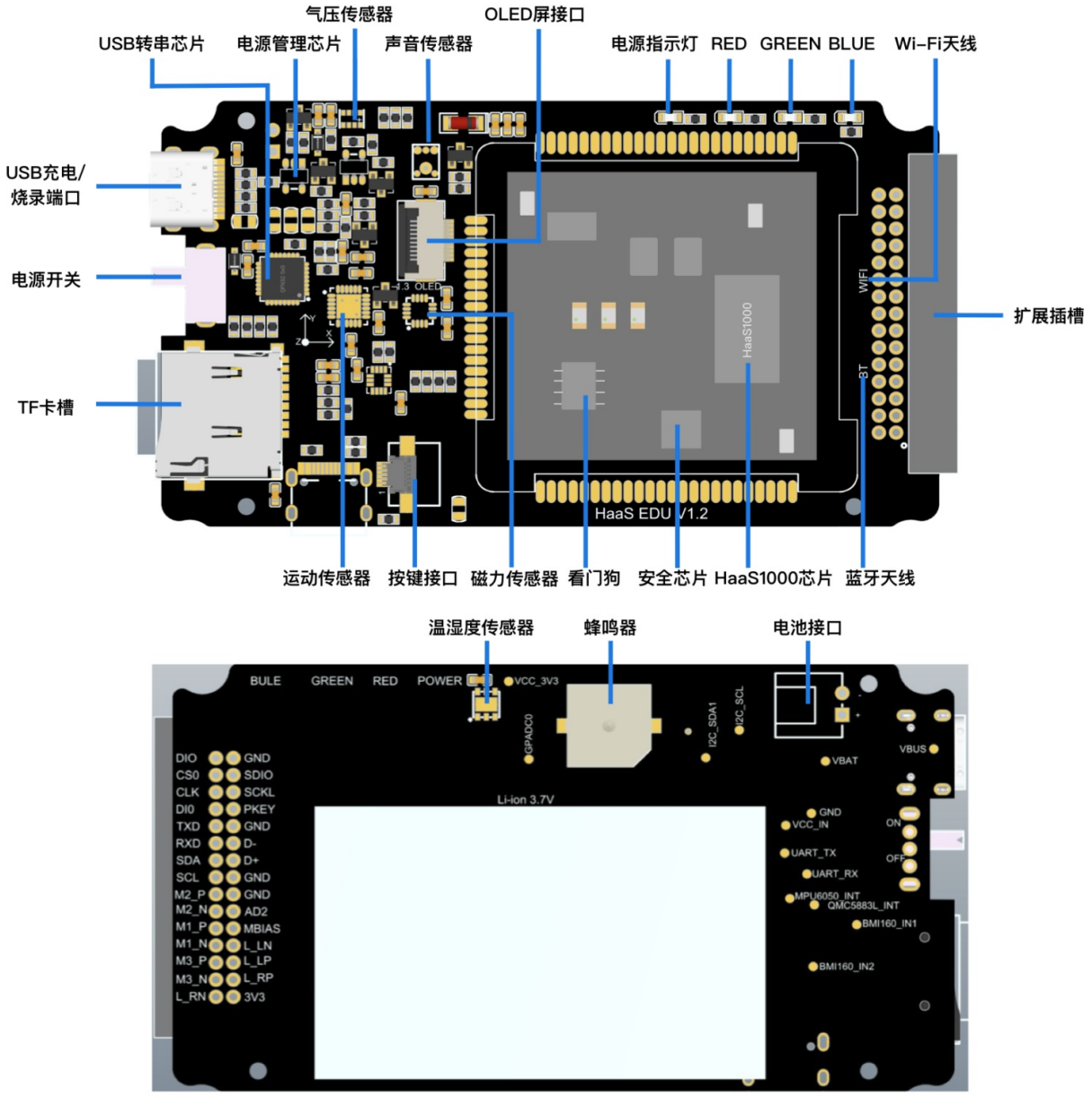


图5 主板功能模块示意图

如图5，HaaS EDU K1主板功能非常丰富，大部分以板载功能呈现，比如：蓝牙/Wi-Fi、OLED屏幕、传感器等，另外还有30PIN扩展接口可以使用。主板尺寸为89.98mm* 49.98mm，充分考虑黄金分割比例，使外形更合理美观，具体板载配置如下表：

名称		描述
CPU	型号	HaaS1000
	架构	Dual Cortex-M33 Dual Cortex-A7

	主频	Cortex-M33 to 300MHz Cortex-A7 to 1GHz
片上Flash		16MB
内存		2.5MB SRAM 16MB PSRAM
加密芯片		Z8IDA
看门狗		ADM706S
蓝牙		蓝牙5.0，支持BLE mesh
Wi-Fi		2.4G/5G双频
串口烧录		FT232RQ
充电管理		1200mAh锂电池 充电电流450mA
加速度传感器*		MPU6050/QMI8610
陀螺仪传感器*		
磁力计*		QMC5883L/QMC6310
气压传感器*		SPL06/QMP6988
温度传感器*		SI7006/CHT8305
湿度传感器		
声音传感器		S08OB383

环境光传感器	AP3216C

4、电气性能

工作电压	3.5 ~ 5V
充电电流	450mA，可充电锂电池
工作温度	-20 ~ 85℃
环境湿度	5~85%RH（无凝结）

> **说明**
>
> ：10月份以后购买的K1批次的硬件略有不同，替换了部分sensor，可以通过软件接口区分，具体为：
>
> MPU6050 --> QMI8610
>
> QMC5883L --> QMC6310
>
> SPL06 --> QMP6988
>
> SI7006 --> CHT8305。

三、软件教程

为了帮助大家能更快地上手物联网相关场景的应用开发，HaaS EDU研发团队为大家精心打造了10大场景式应用打造案例，每一个都是不同的知识点，帮助大家来快速上手常见物联网开发技术的学习。

1、HaaS EDU K1示例操作

在开始代码编写之前，先学习官方提供固件中的示例操作是个不错的注意，可以先将案例体验一遍。

首先，来看一下 HaaS EDU K1的菜单操作。

如上图所示，HaaS EDU K1 提供了：

1、1个1.3寸的信息显示屏

2、5个按键，其中4个可编程按键

3、4个LED，其中3个可编程LED

来完成系统状态的指示以及场景案例的切换，具体的操作为：

1、K1：系统菜单选项左移。在一级菜单时，可切换场景案例到上一个

2、K2：系统菜单中未定义。在有多选项的场景案例中可以切换

3、K3：系统菜单中选项右移。在一级菜单时，可切换场景案例到下一个

4、K4：系统菜单中确认键。进入到当前显示的场景案例

那么如何退出当前的场景案例到上一级菜单呢？可以通过同时按下 “K1+K2”的形式返回上一级菜单。

2、场景式案例介绍

基于HaaS EDU K1 硬件本体，在无须外接任何外设的情况下，官方提供了10个精心打造的场景式案例，每一个都是不同的知识点，先来一个10个案例的总览：

10大场景中包含了常见的物联网传感器、相应的操作案例、开源代码。寓教于乐，在边玩的过程中就能将知识点学习了。场景式案例包括：

- 首页系统信息屏
- 温湿度计
- 陀螺仪小球
- 分歧争端机
- 电子罗盘
- 光照信息屏
- 大气压海拔仪
- 复古八音盒
- 贪吃蛇
- 飞机大战

3、学习指南

我们将打造的一系列基于HaaS EDU K1的案例，上传到HaaS技术社区，大家可以下载学习。如需更多技术支持，可加入钉钉开发者群，或者关注微信公众号。

更多技术与解决方案介绍，请访问阿里云AIoT首页https://iot.aliyun.com/ 。

2.HaaS物联网通用硬件简介

HaaS（Hardware as a service）硬件即服务。通过向用户提供物联网场景中的标准硬件以及嵌入到硬件中的软件驱动及功能模块，为用户提供物联网设备高效开发服务。

3.HaaS100快速开始

简介

硬件

下面是HaaS100的简单介绍， 详细功能和参数可以参考HaaS100的产品说明书。

软件

HaaS100搭载AliOS Things物联网操作系统， 点击查看AliOS Things简介 。

环境

AliOS Things提供两套开发环境：

1、针对初学者，建议参考开发环境一键安装文档，搭建环境、下载代码、编译和烧录等。（使用本方式的开发者可以跳过本文档）

2、针对有经验的嵌入式开发者，AliOS Things的开发环境准备， 点击查看相应的开发环境搭建文档：Linux环境安装；Windows环境安装；macOS环境安装。

快速开始

下载代码

```
git clone https://github.com/alibaba/AliOS-Things.git -b dev_3.1.0_haas
```

对于国内用户，为避免从github下载速度较慢，可以从gitee上下载。

```
git clone https://gitee.com/alios-things/AliOS-Things.git -b dev_3.1.0_haas
```

编译

进入代码的顶层目录如AliOS-Things进行编译。可以直接编译application/example/目录下的demo app，或者自己开发的app。下面以编译helloworld_demo为例。

```
aos make helloworld_demo@haas100 -c config
aos make
```

烧录

准备

给HaaS100插上电源（电源正常的情况下板子右上角LED灯会亮），在Micro USB口插上usb线并连接到烧录主机。等待烧录主机发现新插入的usb设备并识别成串口。

如果您的烧录主机没有自动识别usb并安装驱动， 您可以点击这里 USB转UART驱动 手动下载对应的版本并安装。

识别成功后，可以在计算机管理->设备管理器中发现串口设备。

【备注】如果发现PC无法识别出HaaS板的USB/UART端口，例如"Silicon Labs CP210X USB to UART Bridge (COM5)"，那么可以尝试换个USB线试试，市面上存在部分劣质的USB线可以充电用，但是数据通信异常。建议可以拿线和手机连接，确认下USB线和手机之间的数据通信是否正常。

使用命令行方式烧录

打开命令行工具，进入到代码顶层目录，然后输入命令 `aos upload` ，选择相应的串口即可开始烧录。

```
--- Available ports:
---  1: /dev/cu.Bluetooth-Incoming-Port 'n/a'
---  2: /dev/cu.SLAB_USBtoUART u'CP2102N USB to UART Bridge Controller'
---  3: /dev/cu.usbserial-1410 u'CP2102N USB to UART Bridge Controller'
--- Enter port index or full name: 2
```

烧录成功后，显示“Firmware upload succeed“字样。

烧录成功后，也自动将串口信息保存在当前目录下的.aos_config_burn文件中。如果后续烧录时，串口号有变化，可以删除该文件，再使用 `aos upload` 命令烧录。

如果在烧录期间，提示“Please reboot the board manually“，请按复位键复位开发板。

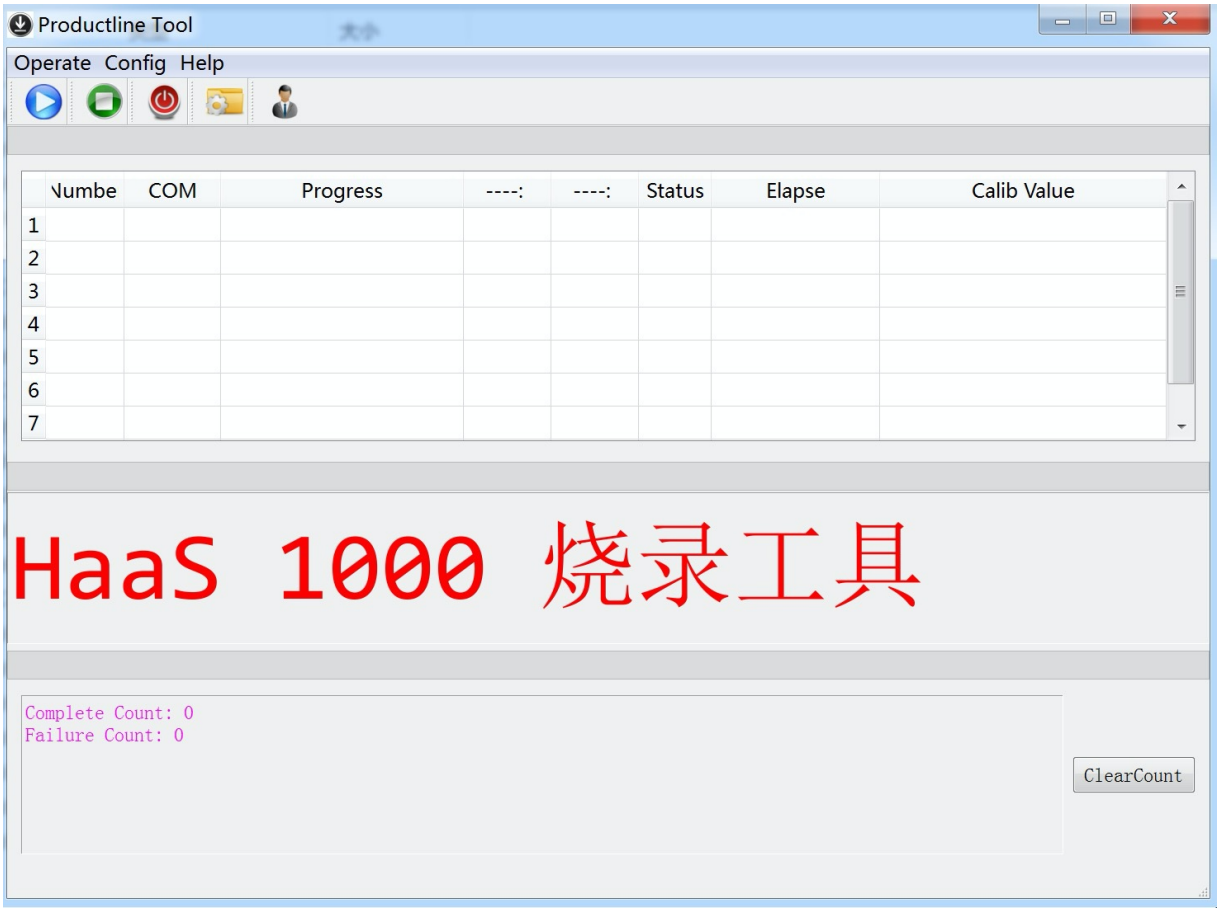
若复位无效，还是一直打印“Please reboot the board manually“，则需要将板子完全断电，再上电，然后重新使用 `aos upload` 命令烧录。

```
Please reboot the board manually.
hello world! count 341
hello world! count 342
```

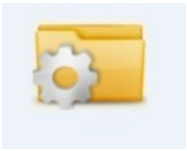
使用GUI工具烧录

第一次编译后从代码顶层目录拷贝platform/mcu/haas1000/release/write_flash_gui目录下的所有文件到windows环境。 后续再编译可以只替换write_flash_gui/ota_bin/ota_rtos.bin即可。

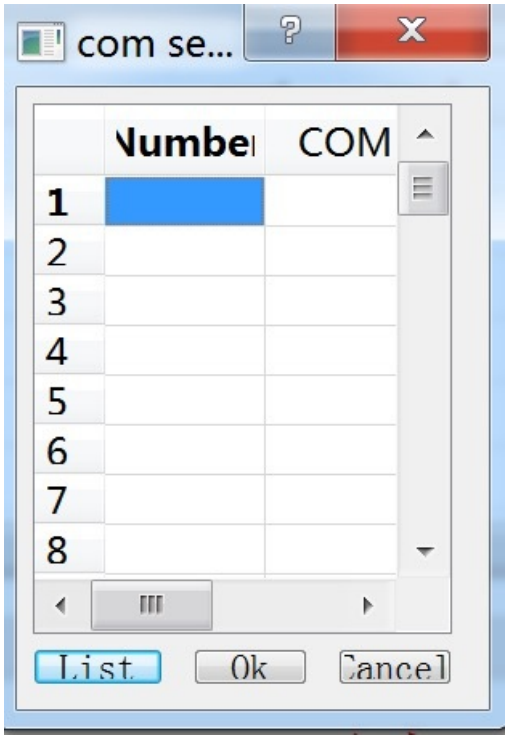
在windows环境下进入write_flash_gui目录，双击运行haas1000_write_flash_main，出现烧录软件的主界面。



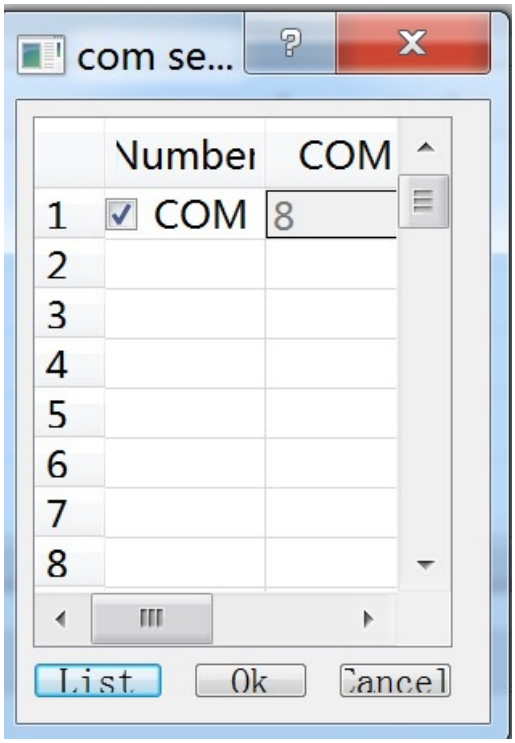
点击Port Config



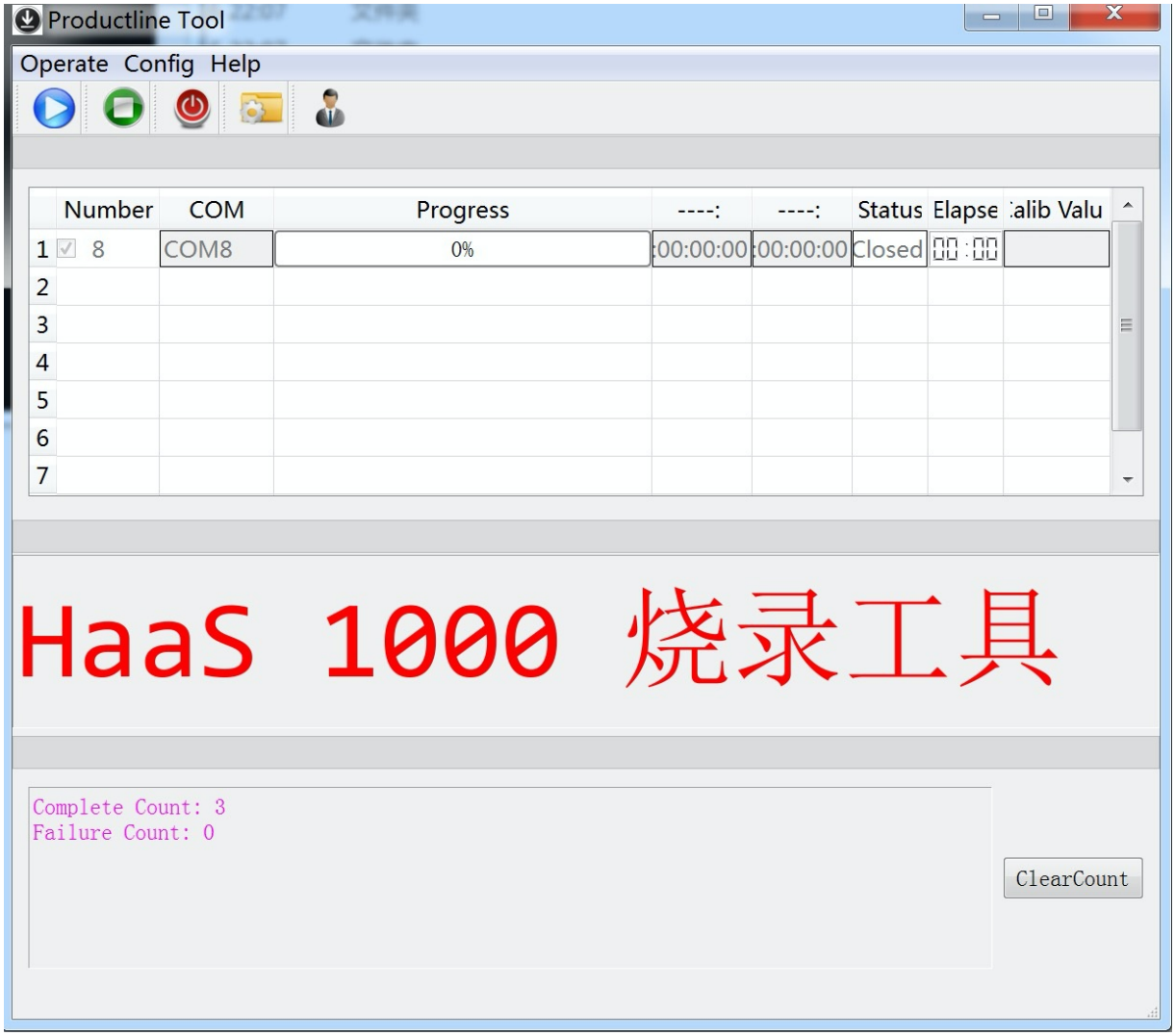
按钮，出现串口配置界面。



点击List按钮，选择HaaS100连接的串口。

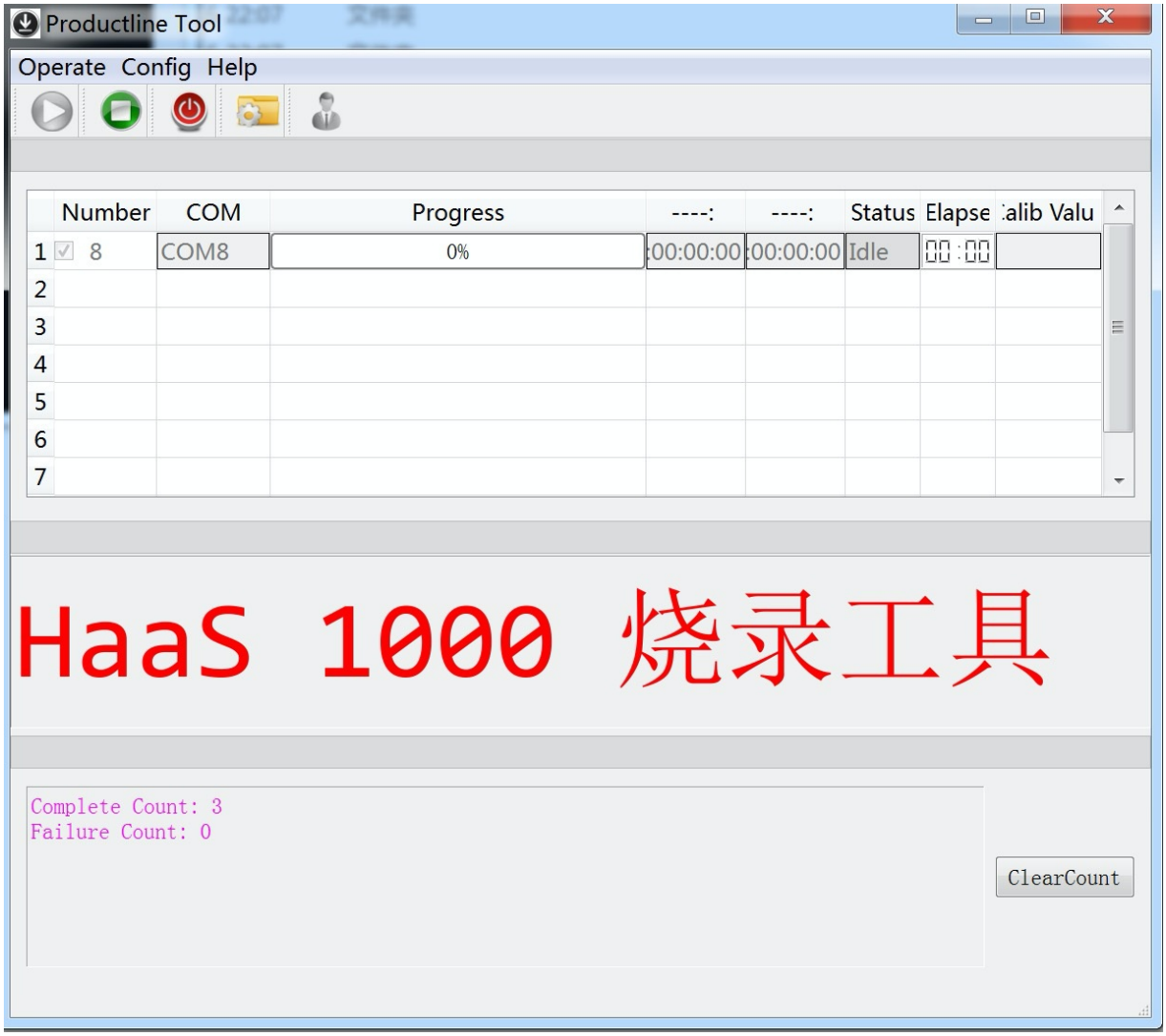


选择后，点击Ok按钮确认关闭串口配置界面，此时主界面如下，烧录状态是Closed。

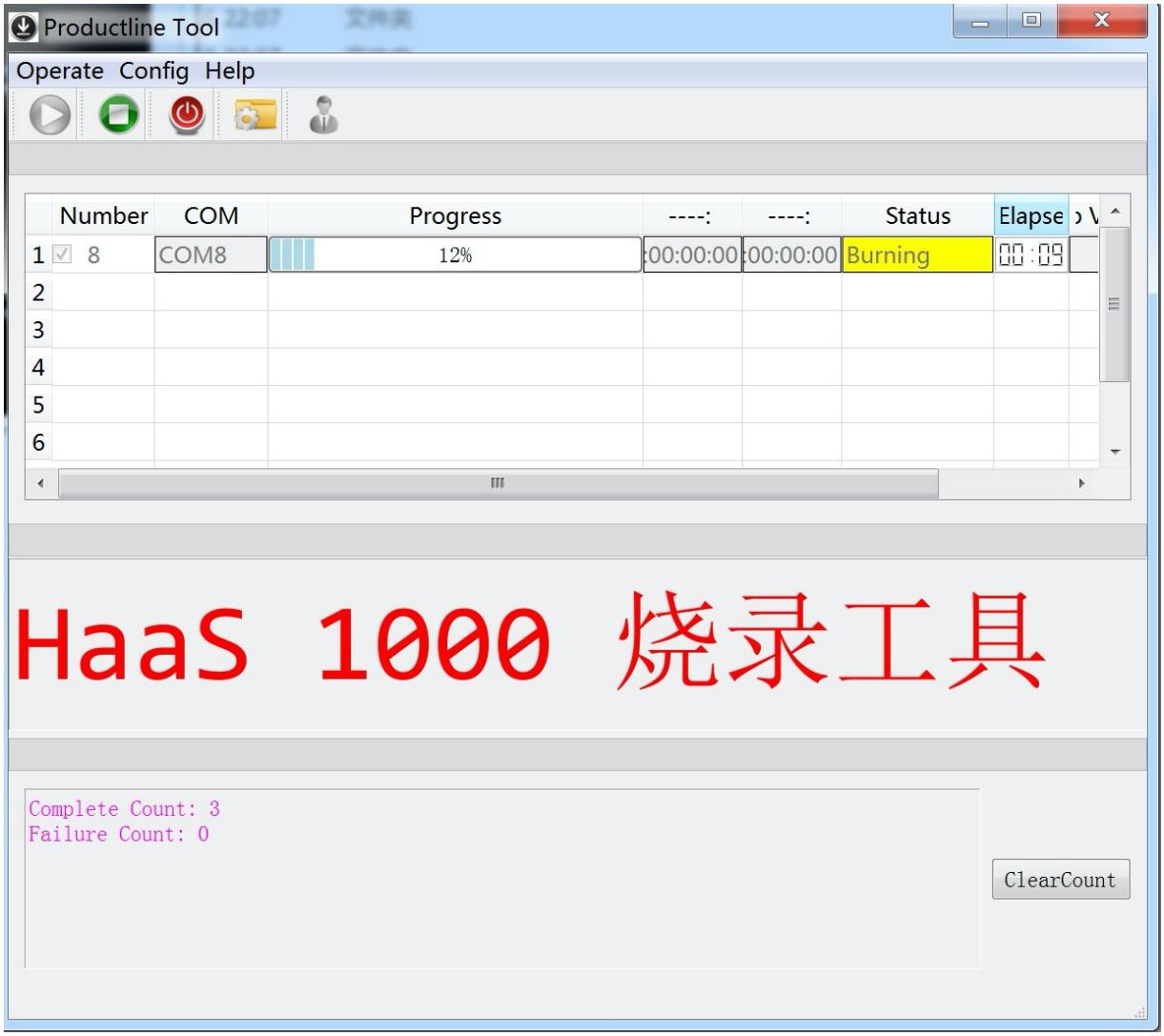


点击Start All

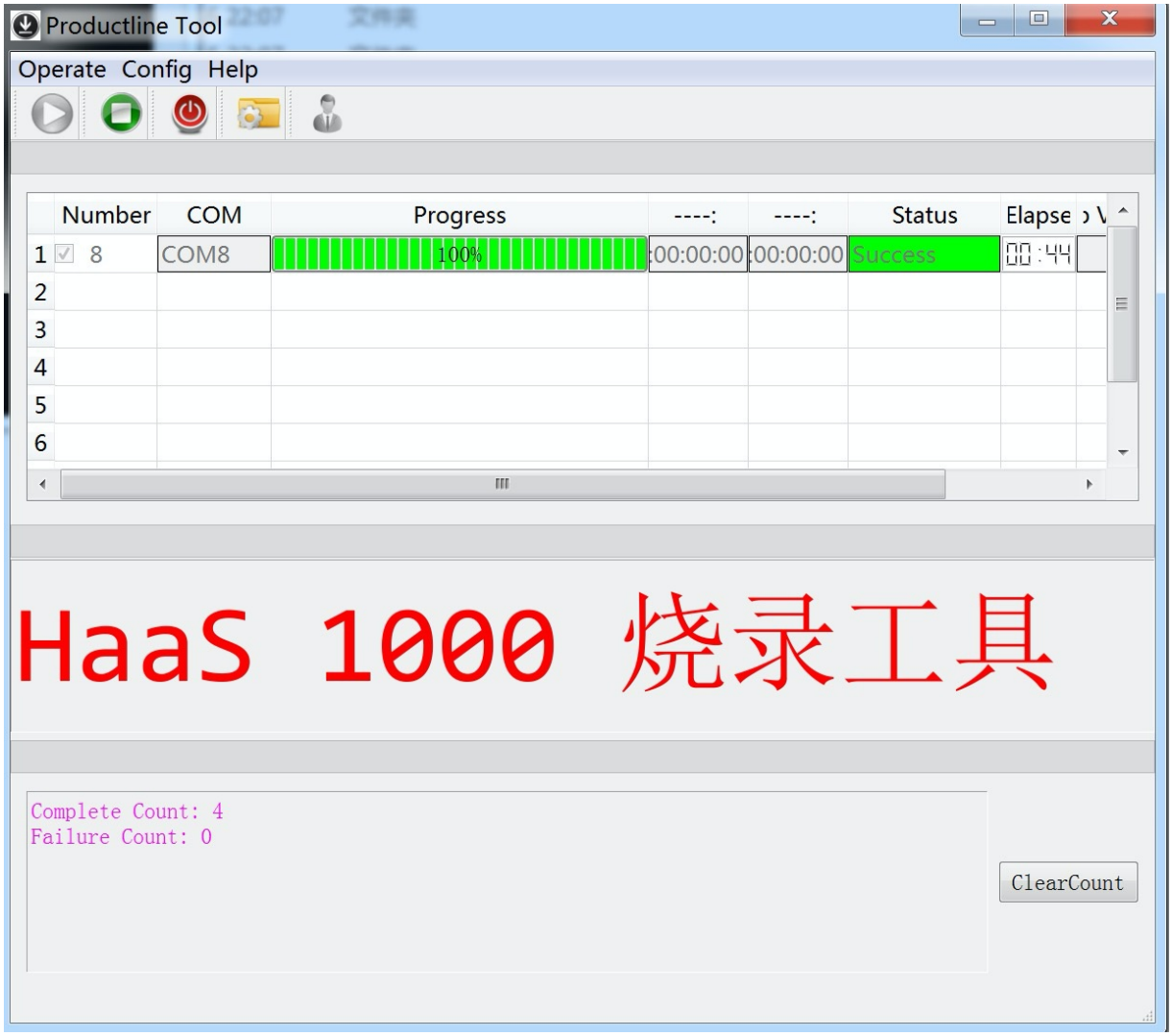
按钮，进入烧录状态，此时烧录状态是Idle，等待设备重启。



短按板子上的重启键或者插拔电源对板子上下电，开始烧录，此时烧录状态是Burning，可以看到烧录的进度。



等待烧录成功，此时烧录的状态的Success，进度是100%。



点击Exit

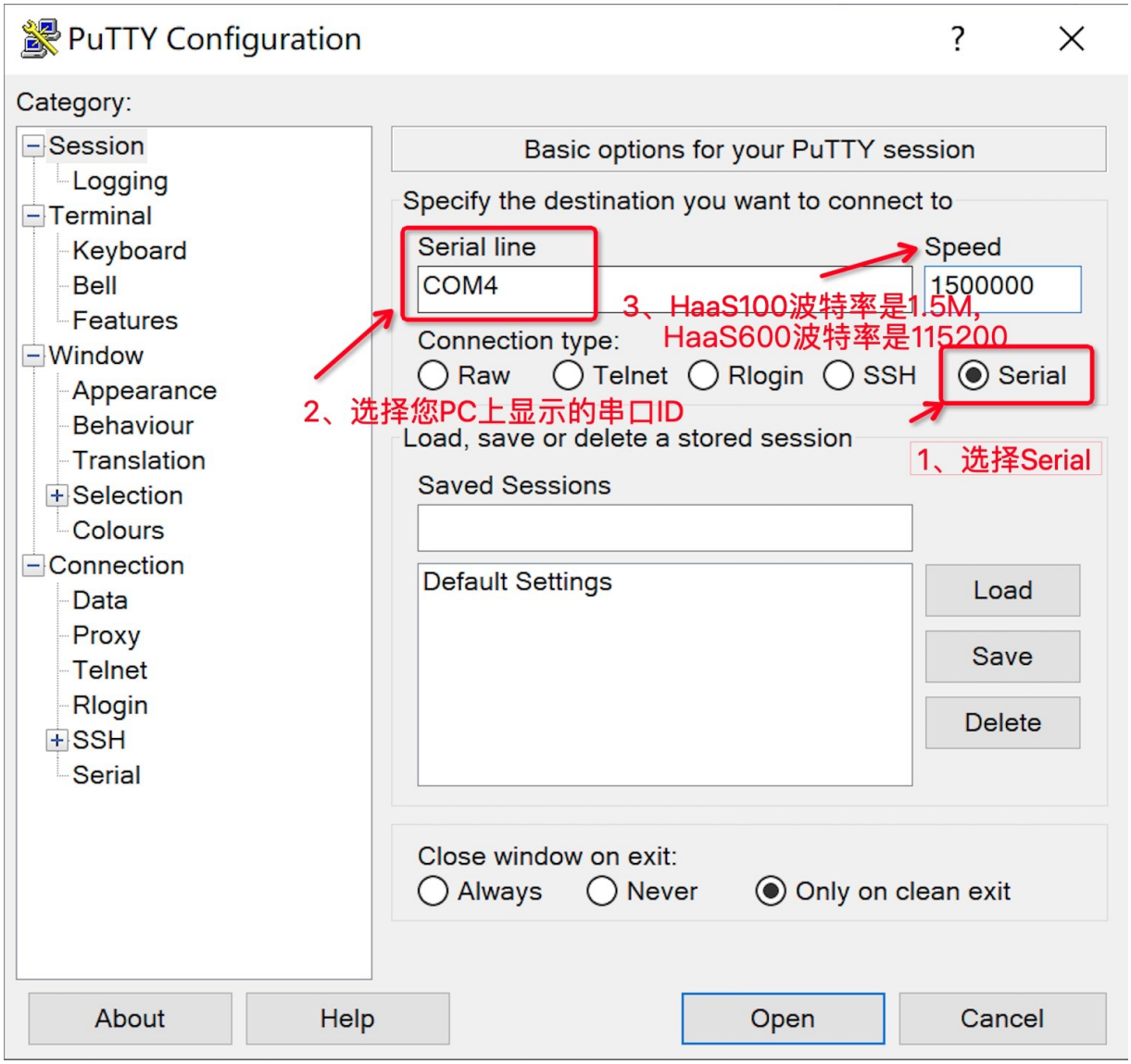
退出即可。

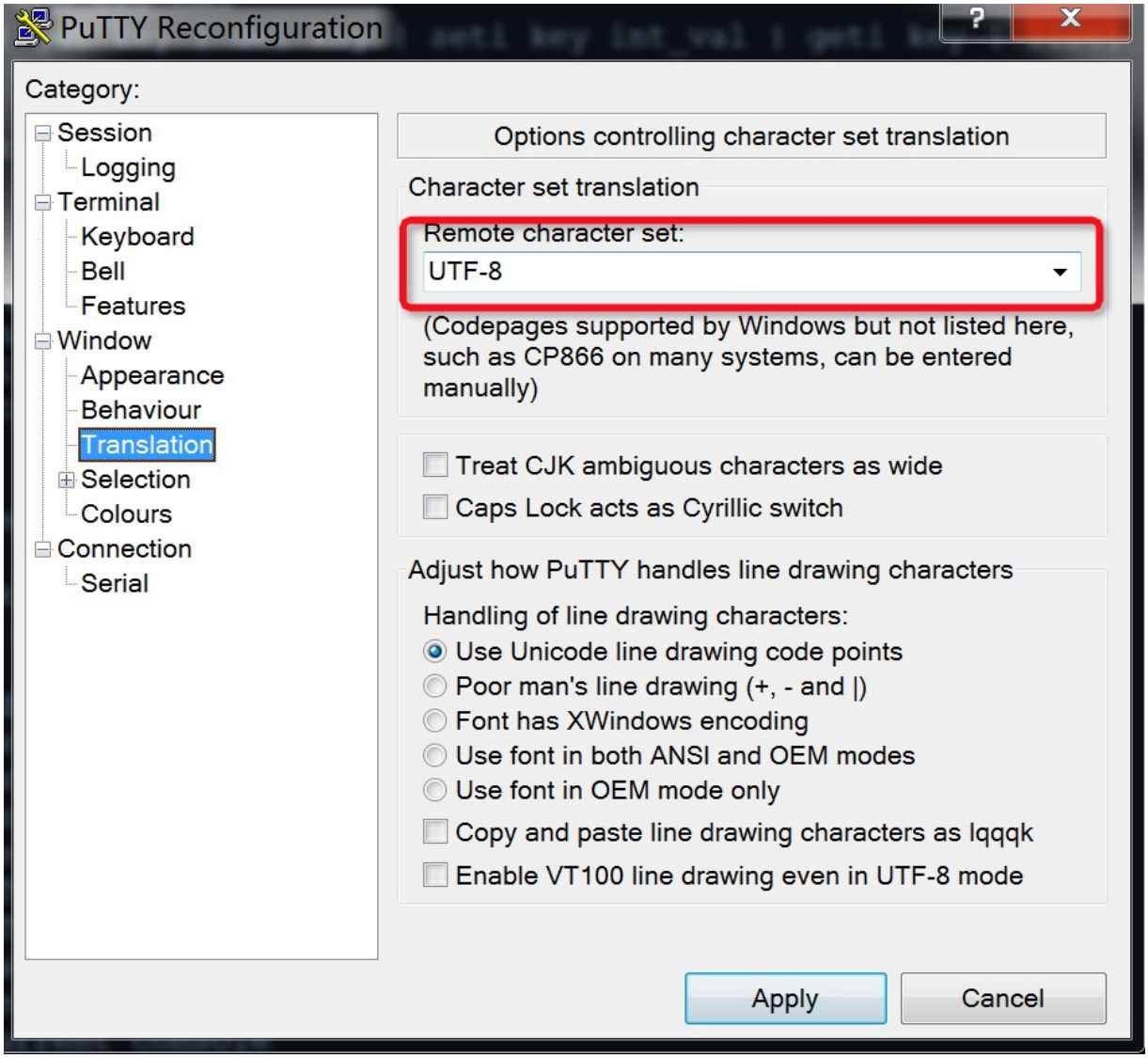
串口打印

使用串口工具如Putty/SecureCRT，设置串口波特率**1500000**。

Putty设置

设置串口波特率1500000，并关闭Parity 和 Flow control，并设置字符编码为UTF-8，否则打印会乱码。


PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Serial
Basic options for your PuTTY session
Specify the destination you want to connect to
Serial line
COM4
Speed
1500000
3、HaaS100波特率是1.5M,
HaaS600波特率是115200
Connection type:
Raw
Telnet
Rlogin
SSH
Serial
2、选择您PC上显示的串口ID
1、选择Serial
Load, save or delete a stored session
Saved Sessions
Default Settings
Load
Save
Delete
Close window on exit:
Always
Never
Only on clean exit
About
Help
Open
Cancel



SecureCRT设置

设置串口波特率1500000，并关闭Parity 和 Flow control， 并设置字符编码为UTF-8，否则打印会乱码。

启动log

设置好串口后， 重启设备就可以看到AliOS Things的启动log，和最后hello world的打印了。

```
     1569/main | app_status_indication_set 1
     1569/main | platform_init_step1 done, user_init=3, ret=0
     1569/main | platform_init_step1 enter temprature cali on signal
     1569/main | board_init platform_init_step1 done
     1573/main_task | sys_init go
     1573/main_task | sys freq calc : 320000000, wifi init 1
     1573/main_task | sys_init soc_peripheral_init done
            Welcome to AliOS Things
     1574/main_task | sys_init aos_components_init done
     1574/main_task |         mesh has been opened
[Jan 01 00:00:01.482]<I>ULOG-test sys_init aos_components_init done

nano entry here!
hello world! count 0
hello world! count 1
hello world! count 2
hello world! count 3
```

如何点亮一个LED灯

上面看过了简单的打印后， 我们做一个简单的点灯实践来练练手。

如下简单修改application/example/helloworld_demo/appdemo.c，增加点灯的代码。

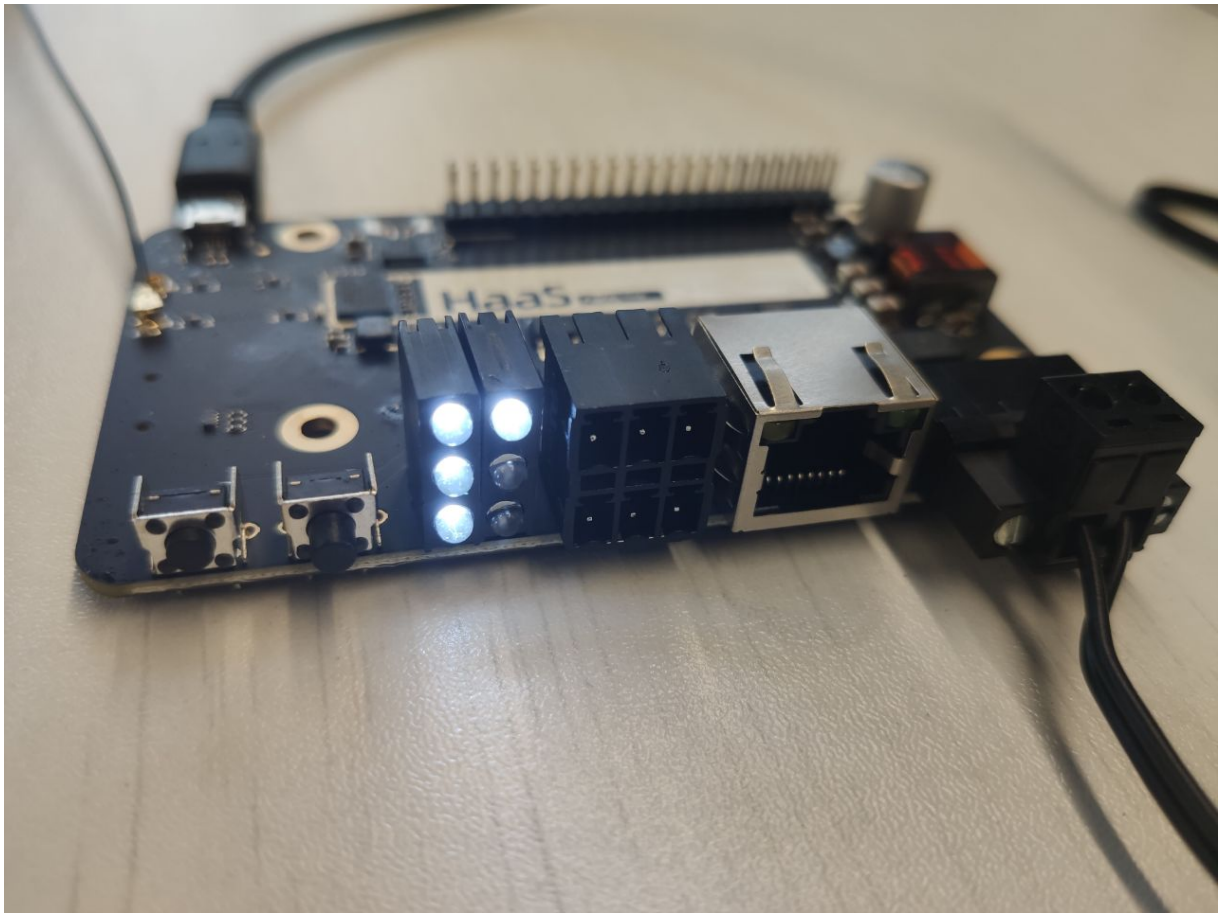
说明：灯的编号，右边从上到下分别是0, 1 , 2；左边从上到下分别是3, 4, 5。 编号0是电源指示灯，不可修改。

```
--- a/application/example/helloworld_demo/appdemo.c
+++ b/application/example/helloworld_demo/appdemo.c
@@ -8,6 +8,7 @@
 #include "aos/init.h"
 #include "board.h"
 #include <k_api.h>
+#include "led.h"

 int application_start(int argc, char *argv[])
 {
@@ -18,6 +19,13 @@ int application_start(int argc, char *argv[])
     //fd = board_lcd_create("name");
     //board_lcd_write(fd,buffer,len);

+    printf("Light all the led switch of the left.\r\n");
+    /* Sleep 1 second to avoid that led switch on/off while the boad boot up. */
+    aos_msleep(1000);
+    led_switch(3,LED_ON);
+    led_switch(4,LED_ON);
+    led_switch(5,LED_ON);
+
     while(1) {
         printf("hello world! count %d \r\n", count++);
```

代码改完后，参考上面的步骤编译烧录，最后启动板子就能看到左边的灯都已经被点亮。



怎么样？ 用HaaS进行开发是不是很简单， 远远不止这些哦，HaaS已经做了很多简化开发的工作， 快来动手探索一下吧。

4.使用VS code IDE开发

`AliOS Studio`是一套基于vscode的开发环境。`AliOS Studio`有以下功能：

- 极佳开发体验、简单操作界面
- 代码补全、索引、提示等
- 编译/下载/调试 AliOS Things
- 适配多种开发板
- 串口工具等

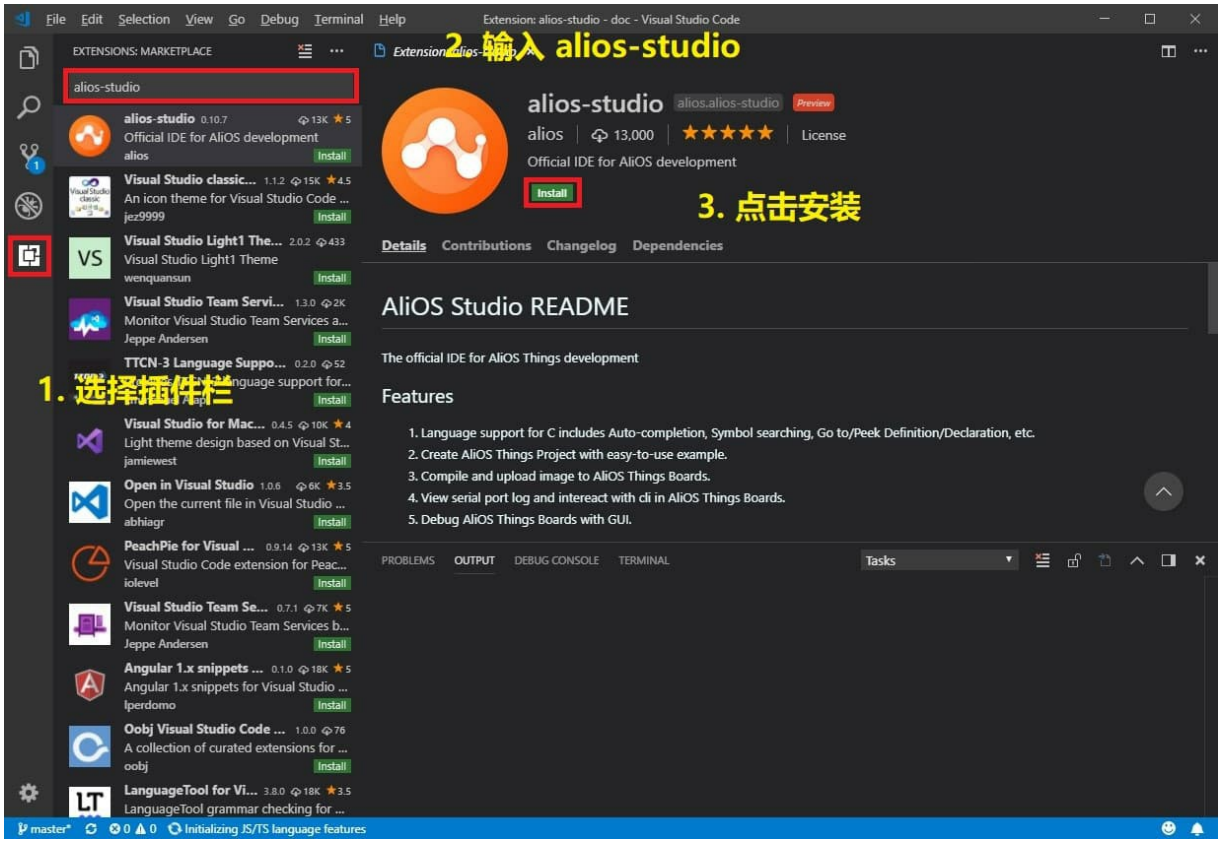
安装

1. 下载/安装 vscode

访问 https://code.visualstudio.com/ 下载并安装`vscode`。

2. 安装AliOS Studio 插件

打开vscode，按照下图所示安装`AliOS Studio`插件：



使用

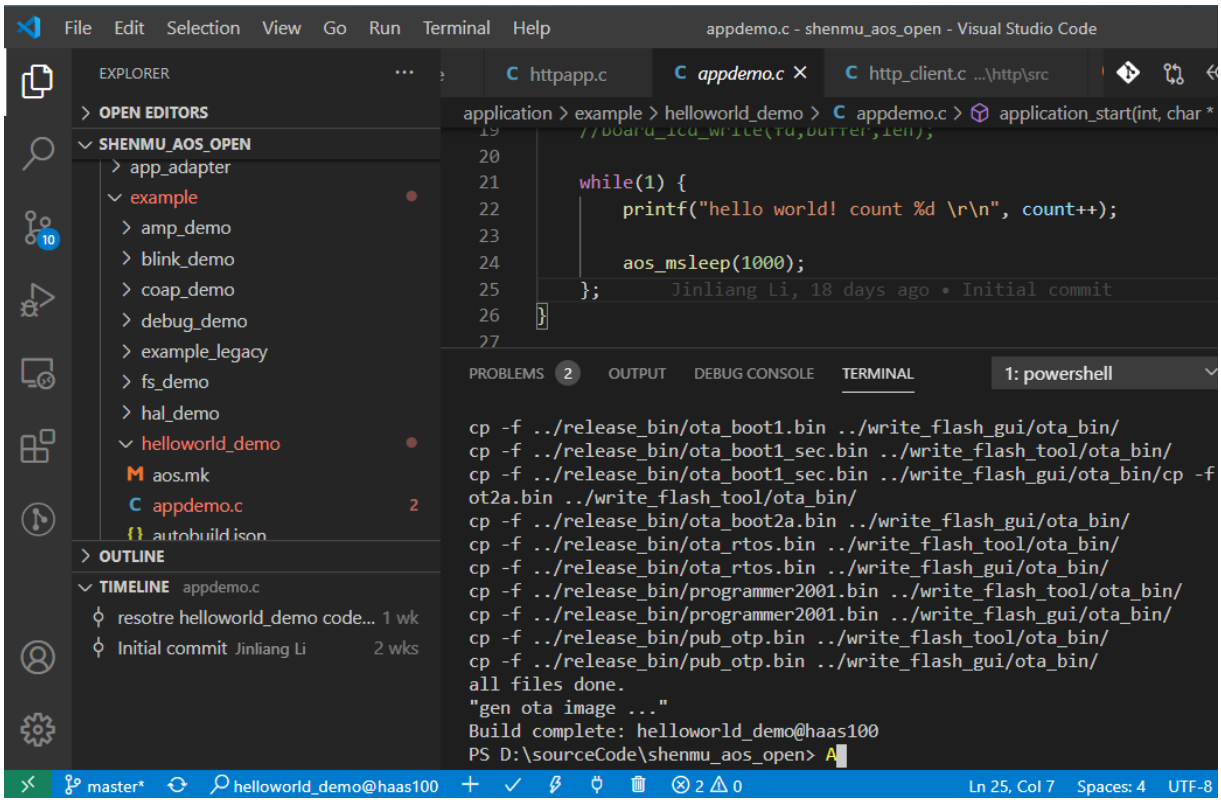
AliOS-Studio 工具栏

`AliOS-Studio` 的主要功能都集中在vscode下方工具栏中，小图标从左至右功能分别是 `创建工程` `编译` `烧录` `串口工具` `清除`

点击左下角“helloworld@developerkit"，即可根据开发要求创建自己的应用以及选择正确的目标板。针对haas板，可以选择一个简单的示例程序 `helloworld_demo` ，开发板 `haas100` 进行验证。

编译 - Build

左侧的 `helloworld@deverloperkit` 是编译目标。如果是基于haas开发板创建的helloworld，此处将显示 `helloworld_demo@haas100` 。点击左下角的“√”进行编译。



烧录

烧录方式1：

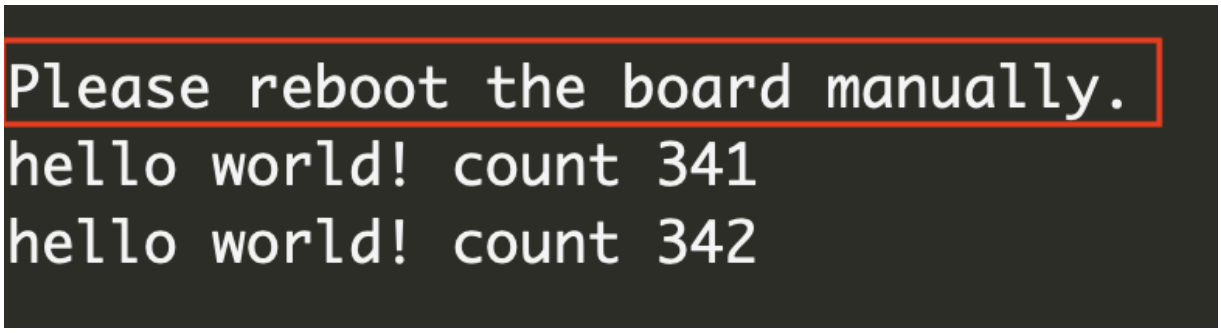
可以直接点击“⚡”标志进行烧录，在随后的终端界面中选择相应的串口即可开始烧录。

```
--- Available ports:
---  1: /dev/cu.Bluetooth-Incoming-Port 'n/a'
---  2: /dev/cu.SLAB_USBtoUART u'CP2102N USB to UART Bridge Controller'
---  3: /dev/cu.usbserial-1410 u'CP2102N USB to UART Bridge Controller'
--- Enter port index or full name: 2
```

烧录成功后，显示“Firmware upload succeed“字样。

烧录成功后，也自动将串口信息保存在当前目录下的.aos_config_burn文件中。如果后续烧录时，串口号有变化，可以删除该文件，再使用点击“⚡☺“标志烧录。

如果在烧录期间，提示“Please reboot the board manually“，请按复位键复位开发板。若复位无效，还是一直打印“Please reboot the board manually“，则需要将板子完全断电，再上电，然后重新点击“⚡☺“标志烧录。



烧录方式2：

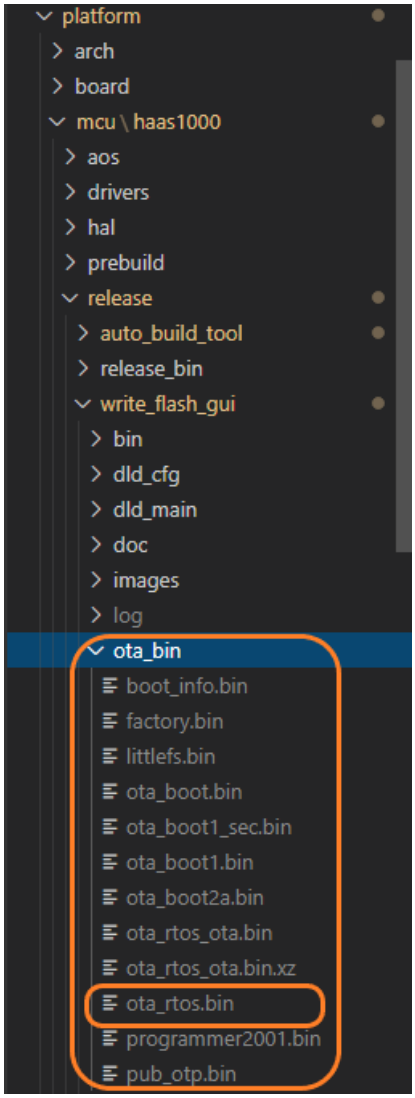
烧写固件至haas开发板，需要使用烧录工具platform/mcu/haas1000/release/write_flash_gui/haas1000_write_flash_main.exe烧写boot程序及应用程序。 其中haas1000为选择的mcu型号。

注意：该烧录工具只能在win环境下运行。 haas100开发板波特率为：1500000。

烧写步骤：

1. 版本文件下载

编译完成后，确认platform/mcu/haas1000/release/write_flash_gui 工具下ota_bin目录的文件齐全，如下图所示。第一次需要下载ota_bin目录下面的所有文件，之后每次烧写只需下载ota_rtos.bin文件即可。



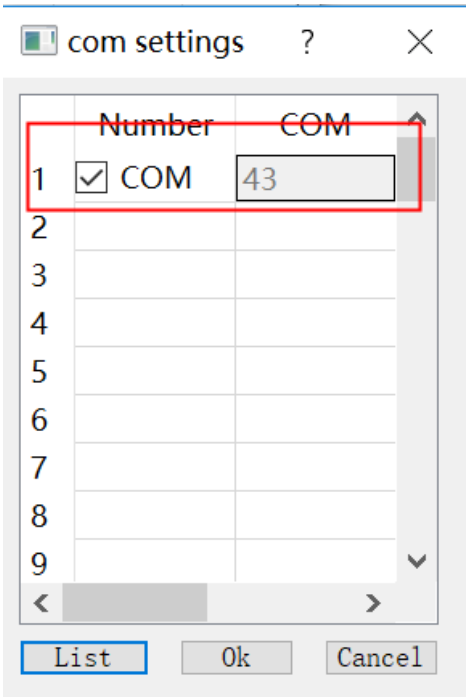
2. 烧写

1. 选择烧写串口 打开haas1000_write_flash_main.exe，点击下图橙色框所示的图标

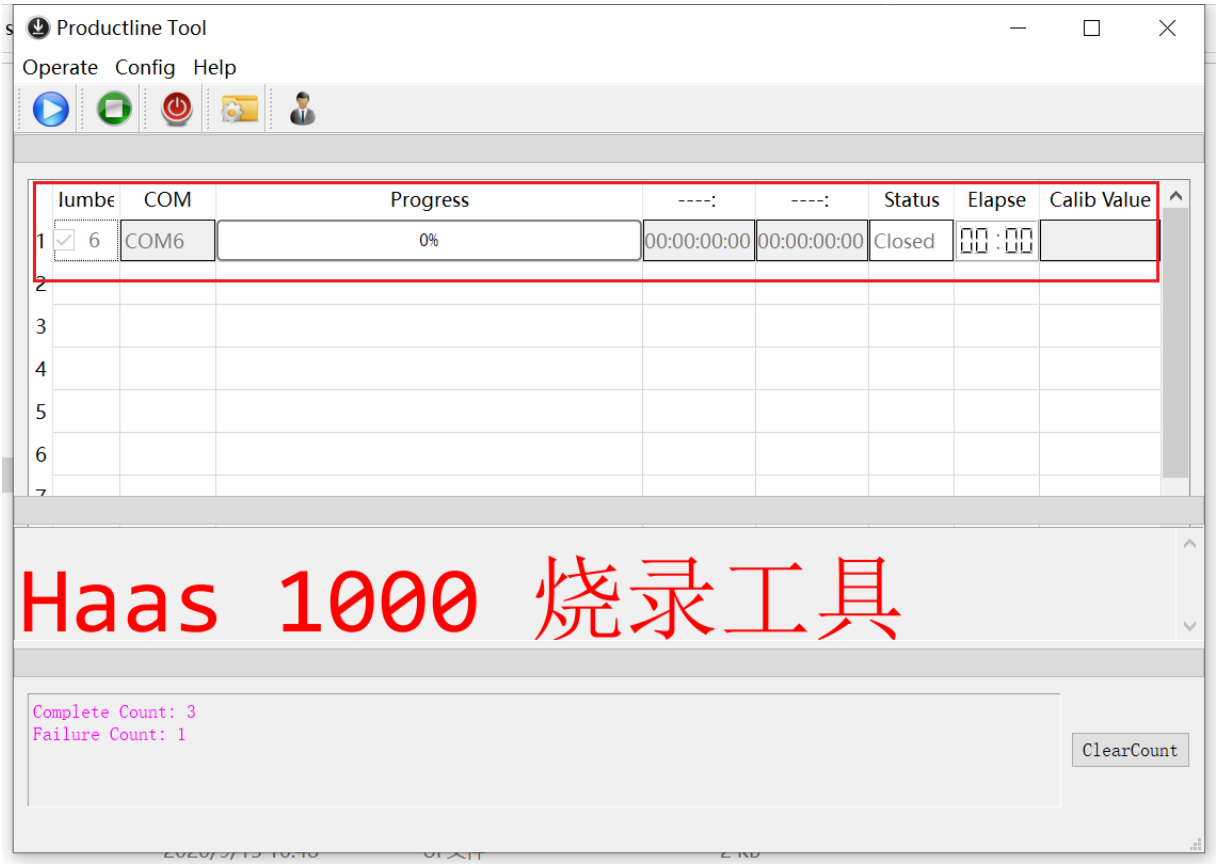
   再点击list，选择串口

   注意：此时需要停止其它使用该串口的工具，否则这里找不到串口。

选择后，图示如下：



点击OK按键，显示如下界面：

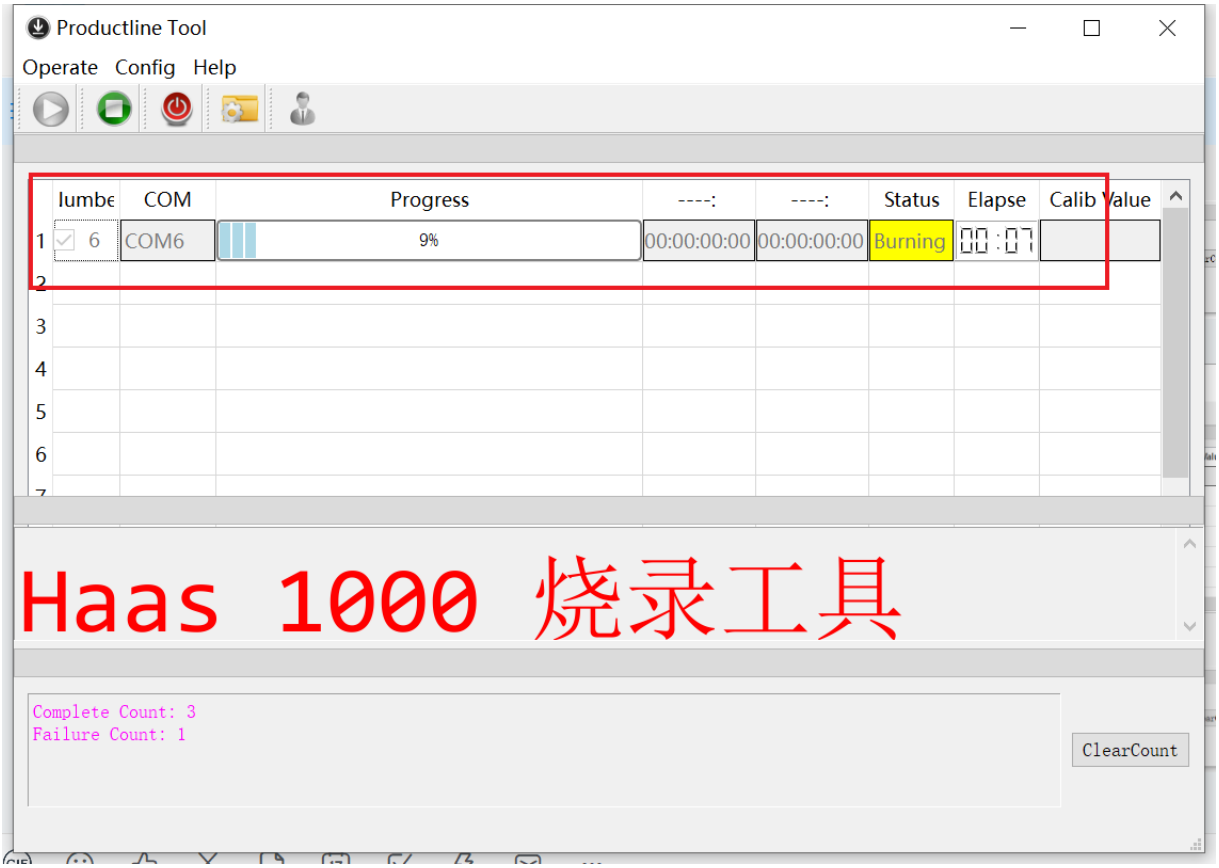


b. 开始烧写点击开始烧写按键

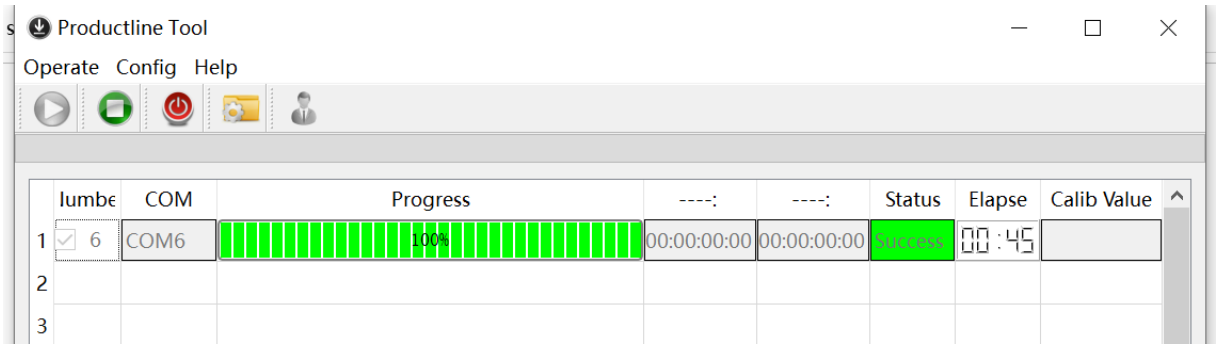
，开始烧写。

c. 重启单板点击蓝色开始烧写按键

后，如果没有开始烧写，需要按复位键（短的按键）经芯片复位，之后，会自动开始烧写。烧写过程界面如下：



烧写完成界面如下：



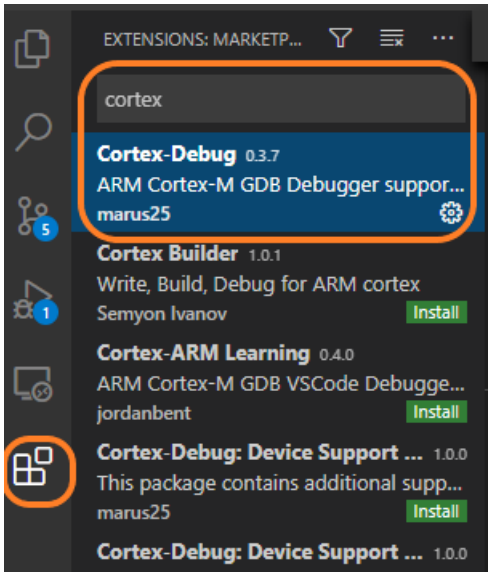
注意：烧写时，有可能烧写失败。烧写失败后，点击绿色停止按钮

，再点击蓝色开始按钮

，重新开始直至成功。d. 烧写完成后，需要退出烧写工具，将板子重新复位。

调试

1. 在 VS code Extension 栏位搜索 Cortex-Debug 并安装, 如图所示。

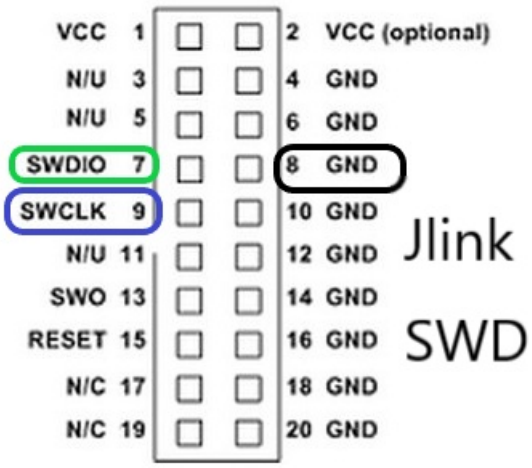


2. 在VS code 打开/.vscode/launch.json文件，设置好debug正确的参数，如下图所示。 PS： 其中executable为ELF可执行文件。armToolchainPath为toolchain的安装目录，serverpath为Jlink GDB的目录，参考

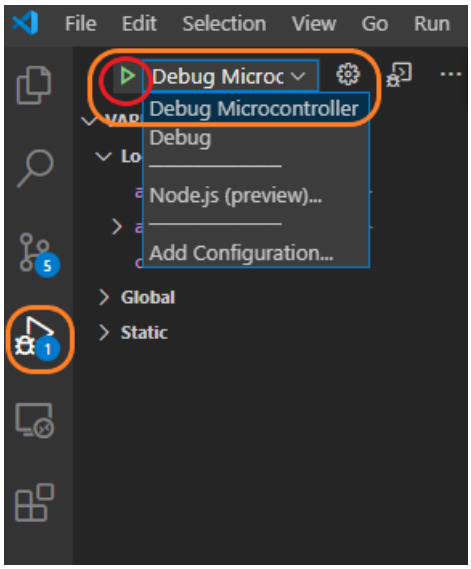
   Windows环境安装

```
.vscode > {} launch.json > Launch Targets > {} Debug
{
    "version": "0.2.0",
    "configurations": [
        {
            "cwd": "${workspaceRoot}",
            "executable": "${workspaceRoot}/out/helloworld_demo@haas100/binary/helloworld_demo@haas100.elf",
            "name": "Debug Microcontroller",
            "request": "launch",
            "type": "cortex-debug",
            "serverpath": "C:\\Program Files (x86)\\SEGGER\\JLink\\JLinkGDBServerCL.exe",
            "servertype": "jlink",
            "device": "Cortex-M4",
            "interface": "swd",
            "armToolchainPath": "C:\\Program Files (x86)\\GNU Tools ARM Embedded\\5.4 2016q3\\bin"
        },
```

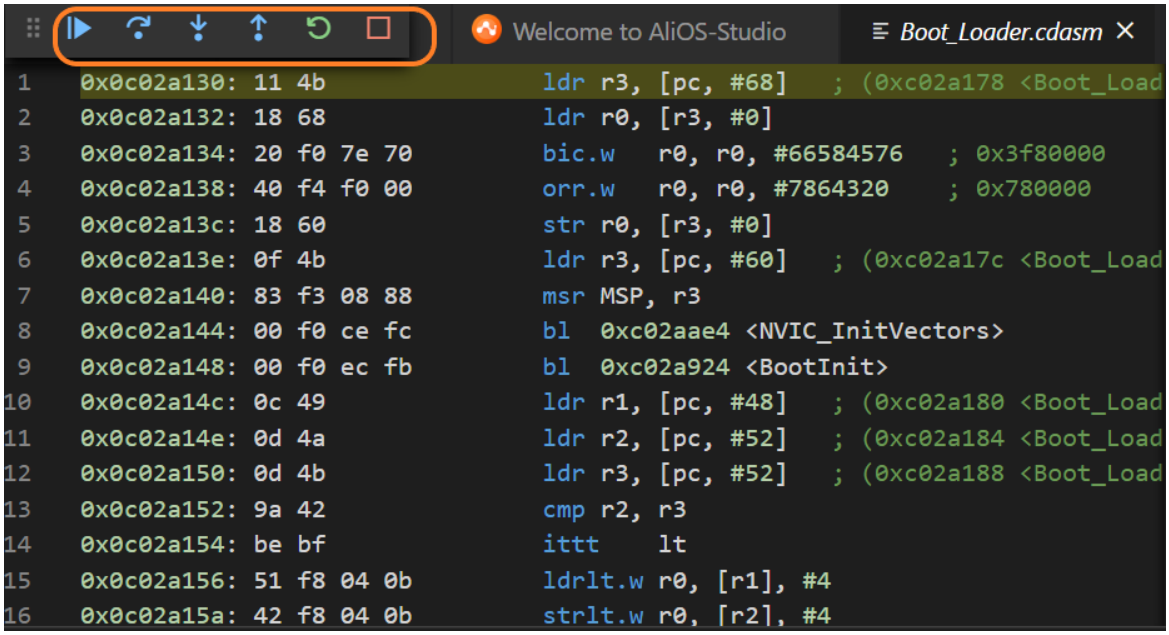
3. 硬件上Jlink连接好开发板,将Haas100板上的SWDIO/SWCLK/GND与Jlink上正确的pin脚连接好。



4. 点击左侧run 按钮，在视窗中选择设置好的名为"Debug Microcontroller"的下拉菜单。

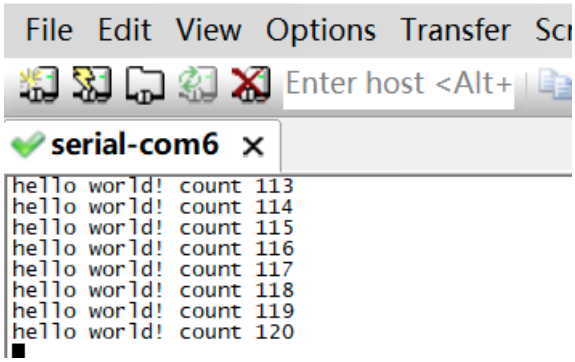


5. Reset Haas板后点击上图红圈所示的执行，即可完成Jlink GDB与开发板的连接。下图所示连接成功。



其中橙框所示的为debug菜单，依次为：Continue/Step over/Step into/Step out/Restart/Stop.

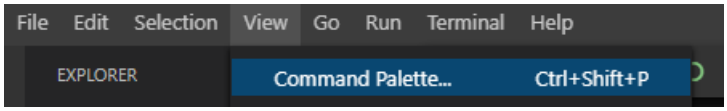
6. 点击continue，可以在设置好的终端上看到helloworld的输出。串口终端的使用可参考《HaaS100快速开始》串口打印章节。



7. 根据实际需要，可以设置断点，单步等各种debug手段。

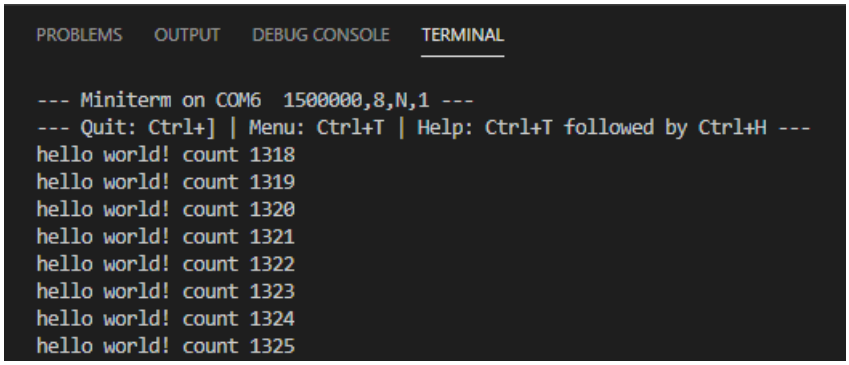
串口监控 - Monitor

1. 通过 USB Micro 线缆连接好开发板和电脑
2. 点击VS code 工具栏View->Commnad palette, 如下如所示：



3. 在键入框输入 connect device. 然后在根据菜单选择COM口以及填写正确的波特率。

4. 连接成功后在terminal显示输出信息，参考下图。



此外，也可以直接打开任意超级终端，设置正确的波特率和串口，可以在终端上看到应用的输出。

5.HaaS 100 硬件规格

硬件配置

类别		参数
CPU	型号	HaaS 1000
	架构	Cortex M33
	主频	300MHz
片上Flash		16MB
内存		2.5MB SRAM 16MB PSRAM

硬件接口

类别	数量	性能指标
TF 卡槽	1个	最大支持 64GB

RS485	1路	波特率支持1200bps ~ 115200bps
RS232	1路	波特率最高支持230400bps，兼容调试串口
以太网（RJ45）	1个	10/100Mbps
按键	2个	复位按键、功能按键
Wi-Fi	1个	2.4G/5G，支持a/b/g/n，1x1
蓝牙	1个	BT5.0/BLE5.0
指示灯	6个	1x电源、5x自定义

电气性能

工作电压	9V~24V
电源保护	具备雷击浪涌保护、反接保护
防护	雷击浪涌 2KV，静电接触 6KV
工作温度	-20~75℃
环境湿度	5~85%RH（无凝结）
运行能力	支持 7*24h 不间断工作

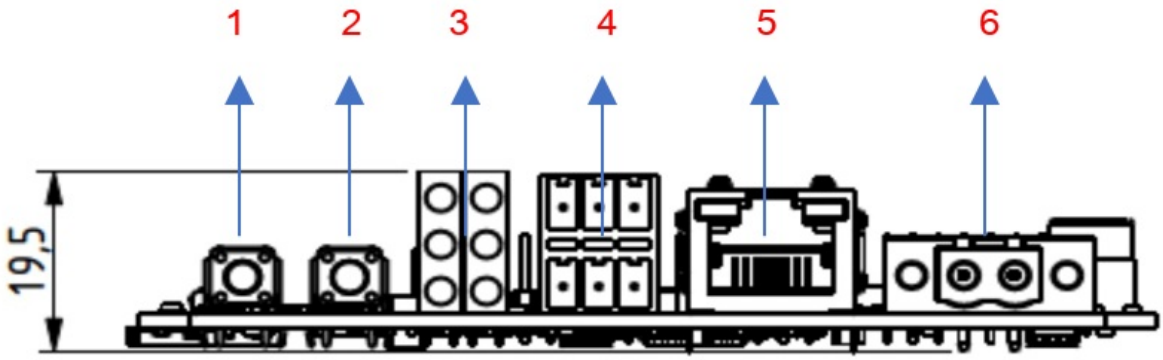
装箱清单

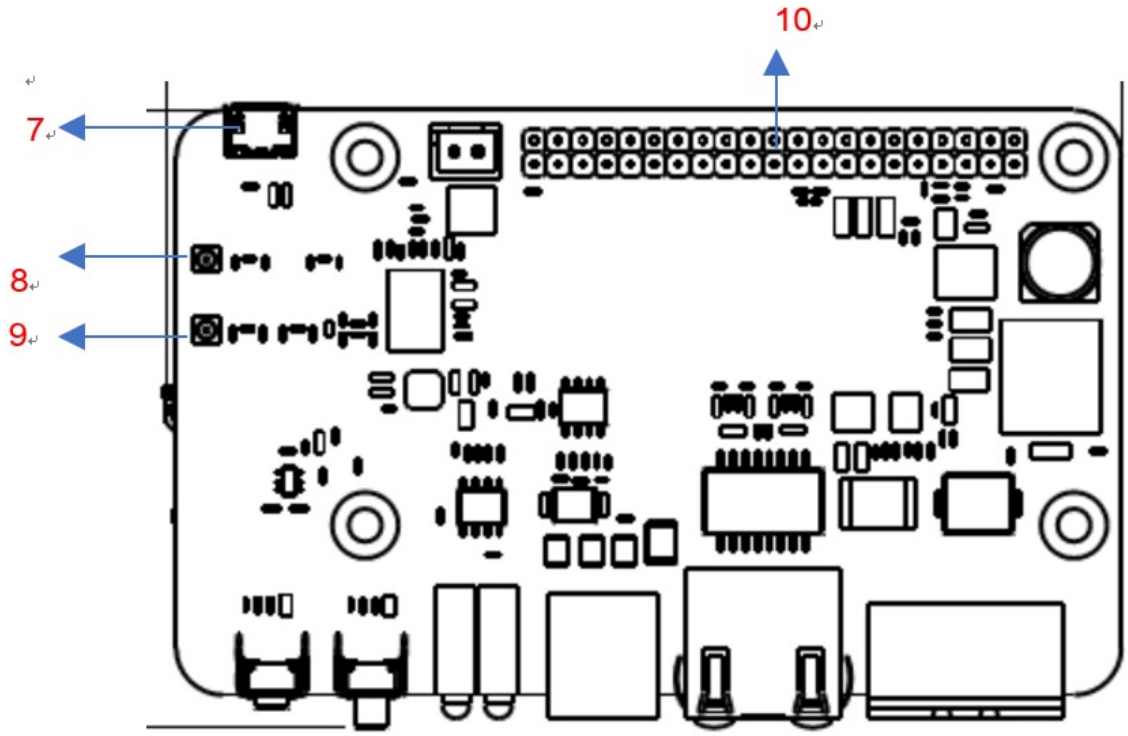
名称	型号/描述	个数
PCBA	HaaS 100	1
说明书	电子版（二维码扫描）	1

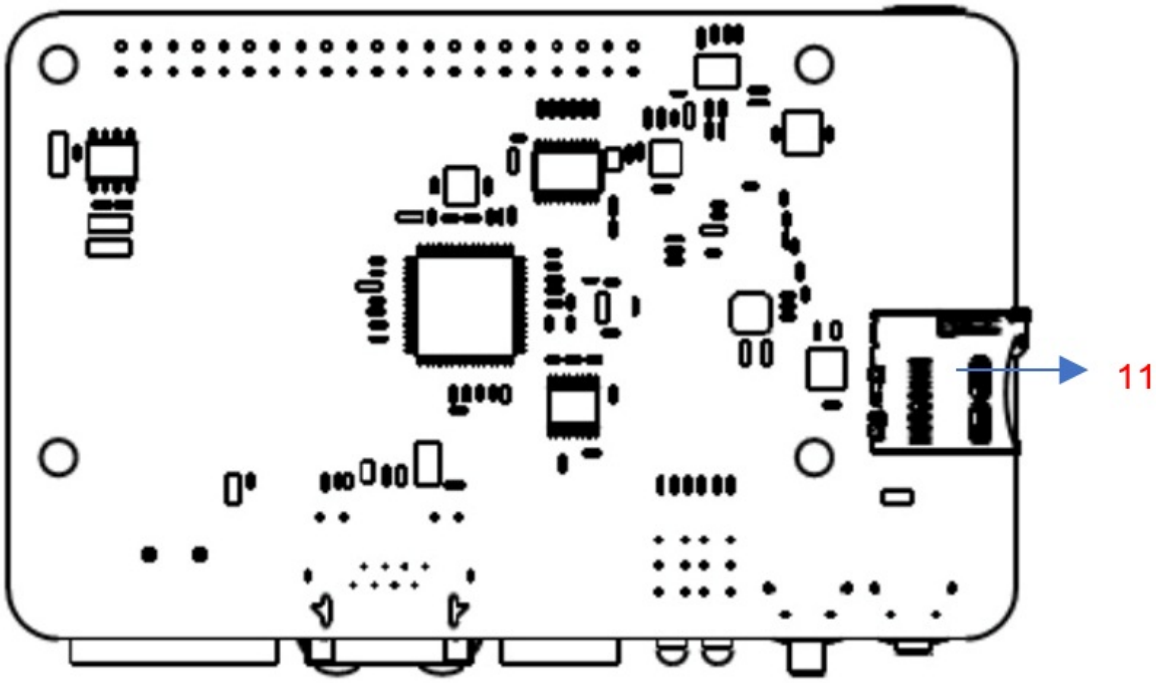
合格证		1
Wi-Fi天线		1
蓝牙天线		1
电源适配器（含凤凰端子）		1
凤凰端子		1

接口定义

主控芯片和接口定义PDF文档下载



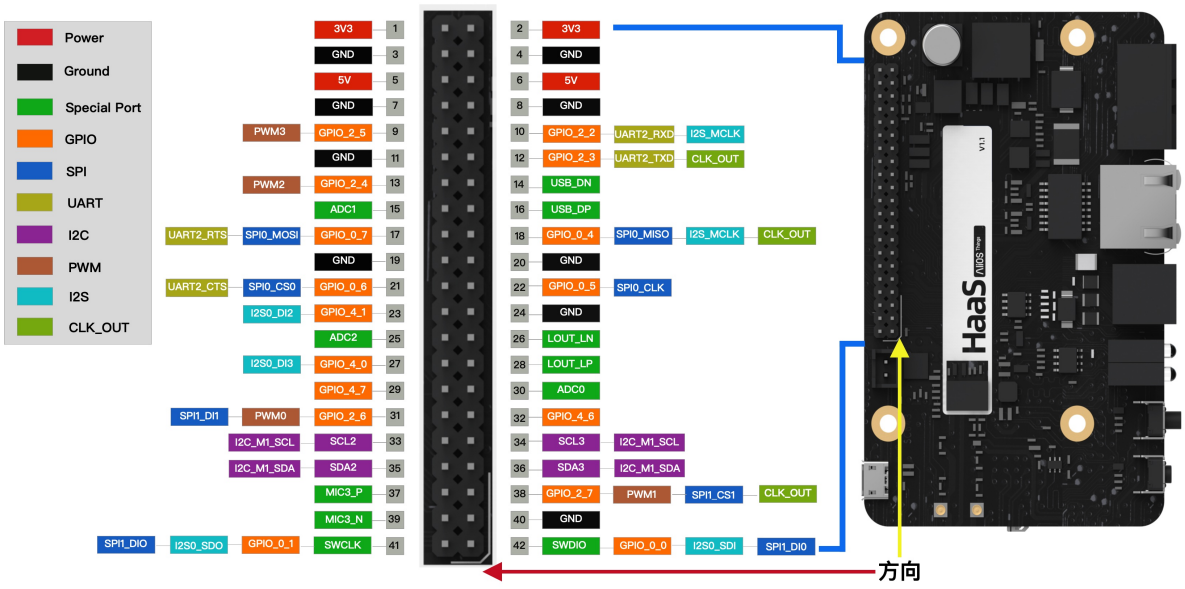




编号	功能	数量	备注
1	复位按键	1	短按复位

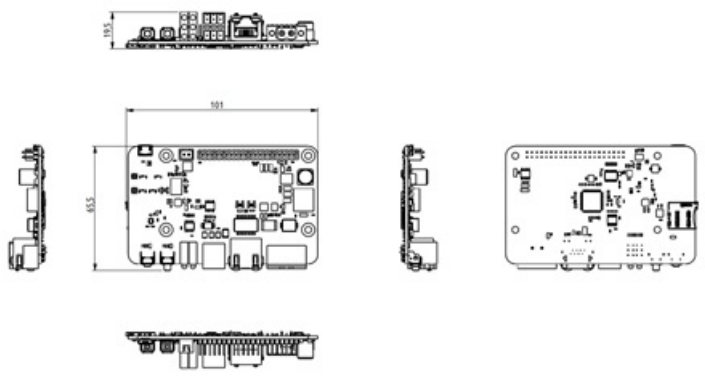
2	功能按键	1	自定义功能按键
3	指示灯	6	左上为电源指示灯，其他为自定义灯
4	RS-485	1	上方是485接口，左中右对应A/B/G
	RS-232	1	下方是232接口，左中右对应R/T/G
5	以太网	1	10/100MBps
6	电源	1	9-24V 直流电源
7	Micro-USB	1	烧录口
8	蓝牙天线	1	2.4G
9	WIFI天线	1	2.4G/5G
10	40Pin扩展槽	1	具体看下图定义
11	TF 卡槽	1	最高支持64GB

排针接口



GPIO在代码中的对应表				
GPIO名称	软件代码宏定义		GPIO名称	软件代码宏定义
GPIO_0_0	HAL_GPIO_PIN_P0_0		GPIO_2_4	HAL_GPIO_PIN_P2_4
GPIO_0_1	HAL_GPIO_PIN_P0_1		GPIO_2_5	HAL_GPIO_PIN_P2_5
GPIO_0_2	HAL_GPIO_PIN_P0_2		GPIO_2_6	HAL_GPIO_PIN_P2_6
GPIO_0_3	HAL_GPIO_PIN_P0_3		GPIO_2_7	HAL_GPIO_PIN_P2_7
GPIO_0_4	HAL_GPIO_PIN_P0_4		GPIO_3_0	HAL_GPIO_PIN_P3_0
GPIO_0_5	HAL_GPIO_PIN_P0_5		GPIO_3_1	HAL_GPIO_PIN_P3_1
GPIO_0_6	HAL_GPIO_PIN_P0_6		GPIO_3_2	HAL_GPIO_PIN_P3_2
GPIO_0_7	HAL_GPIO_PIN_P0_7		GPIO_3_3	HAL_GPIO_PIN_P3_3
GPIO_1_0	HAL_GPIO_PIN_P1_0		GPIO_3_4	HAL_GPIO_PIN_P3_4
GPIO_1_1	HAL_GPIO_PIN_P1_1		GPIO_3_5	HAL_GPIO_PIN_P3_5
GPIO_1_2	HAL_GPIO_PIN_P1_2		GPIO_3_6	HAL_GPIO_PIN_P3_6
GPIO_1_3	HAL_GPIO_PIN_P1_3		GPIO_3_7	HAL_GPIO_PIN_P3_7
GPIO_1_4	HAL_GPIO_PIN_P1_4		GPIO_4_0	HAL_GPIO_PIN_P4_0
GPIO_1_5	HAL_GPIO_PIN_P1_5		GPIO_4_1	HAL_GPIO_PIN_P4_1
GPIO_1_6	HAL_GPIO_PIN_P1_6		GPIO_4_2	HAL_GPIO_PIN_P4_2
GPIO_1_7	HAL_GPIO_PIN_P1_7		GPIO_4_3	HAL_GPIO_PIN_P4_3
GPIO_2_0	HAL_GPIO_PIN_P2_0		GPIO_4_4	HAL_GPIO_PIN_P4_4
GPIO_2_1	HAL_GPIO_PIN_P2_1		GPIO_4_5	HAL_GPIO_PIN_P4_5
GPIO_2_2	HAL_GPIO_PIN_P2_2		GPIO_4_6	HAL_GPIO_PIN_P4_6
GPIO_2_3	HAL_GPIO_PIN_P2_3		GPIO_4_7	HAL_GPIO_PIN_P4_7

外形尺寸



HaaS100 PCBA 2D参考图下载（dxf格式）

主控芯片HaaS1000规格书下载（PDF)

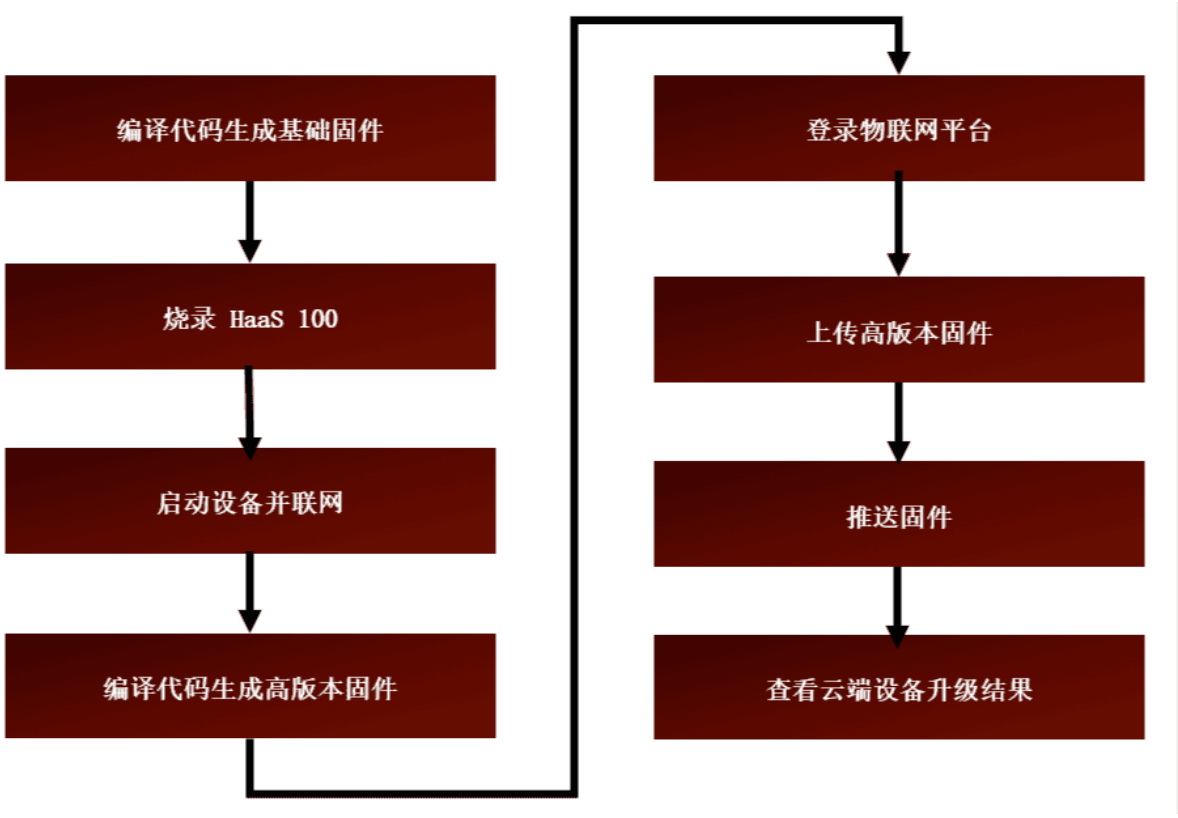
HaaS100测试报告（PDF）

6.应用笔记

6.1. HaaS100 OTA使用手册

概述

HaaS 100是一款针对IOT场景的公板，除支持前面介绍的功能外，也支持OTA功能。OTA为云端一体化技术，HaaS 100 OTA 对接的云端为阿里云物联网平台，设备端搭载的是AliOS Things OTA技术，主要功能列表如下： 1.支持乒乓升级：固件可在两个分区运行，支持固件版本回退，保证设备安全不变砖； 2.支持断点续传：弱网环境下，支持固件从断点处继续下载； 3.支持固件验签：固件可在云端或用本地签名工具进行数字签名，设备端可完成对固件验签； 4.支持https下载方式：除支持http下载外，支持https下载方式； 5.支持MD5/SHA256固件完整性检验：为保证固件完成性，固件下载完成后，都有完整性校验； 6.支持网关子设备升级：当HaaS 100 做网关时，HaaS 100 OTA 除支持网关本身的升级外，也支持其子设备的升级； 升级操作流程如下：



下面以ota_demo为例介绍HaaS 100的OTA如何使用；

使用流程

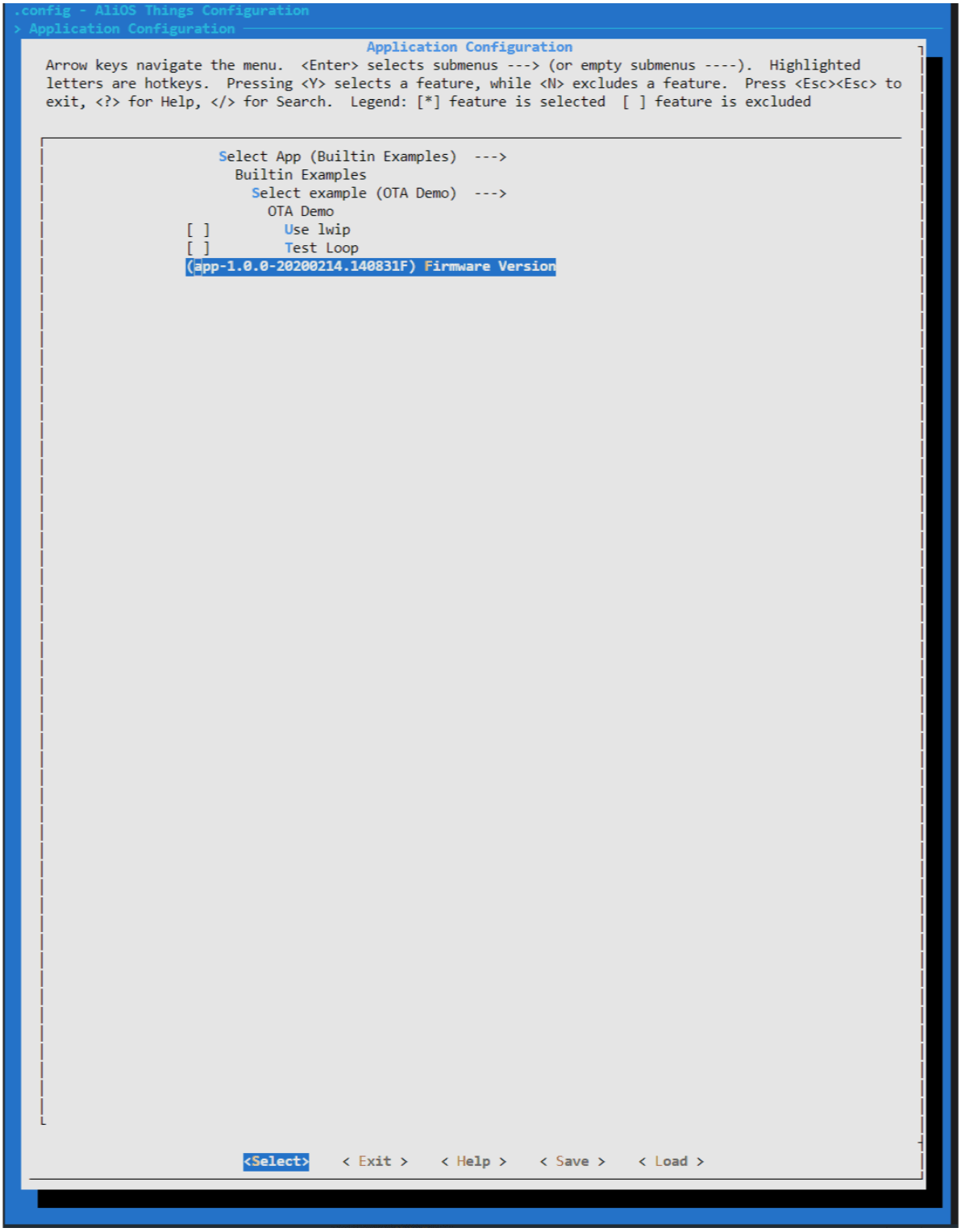
HaaS 100 搭载的是AliOS Things物联网操作系统，编译环境支持windows和linux,下面以linux环境为主介绍使用过程：

- **1.选择app和board**

输入命令：

```
# 清除之前配置
$ aos make distclean
# 配置app为ota_demo，board为haas100
$ aos make ota_demo@haas100 -c config
```

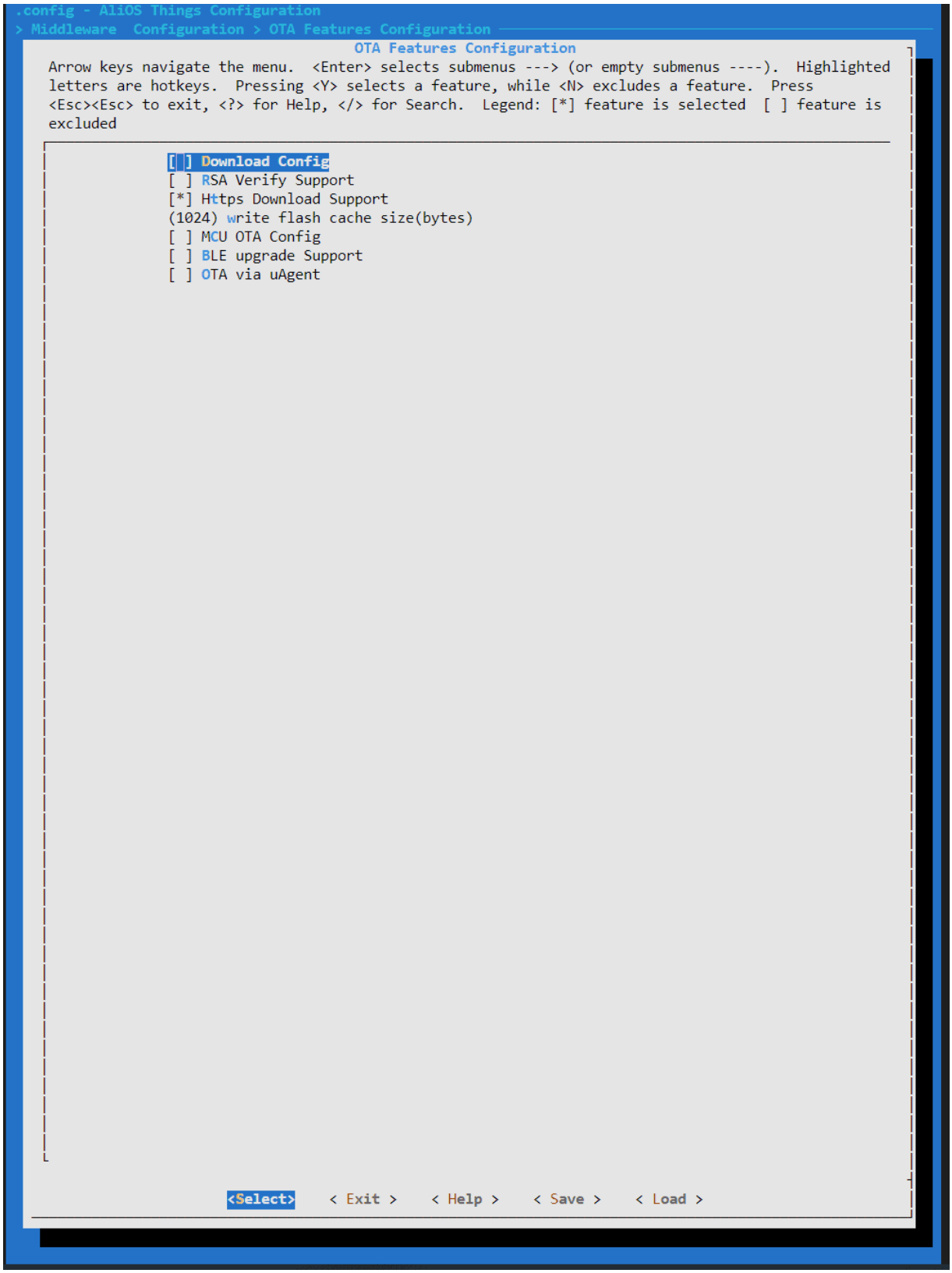
- **2.配置固件版本号** 输入命令：aos make menuconfig, 如下图： 选择顺序：`Application Configuration -> Firmware Version` 根据需求修改版本号。



- **3.选择OTA组件及功能**

完成步骤二，返回主界面： 选择顺序：

```
Middleware  Configuration  --->-*- OTA Features  --->OTA Features Configuration
```



相关功能介绍如下：

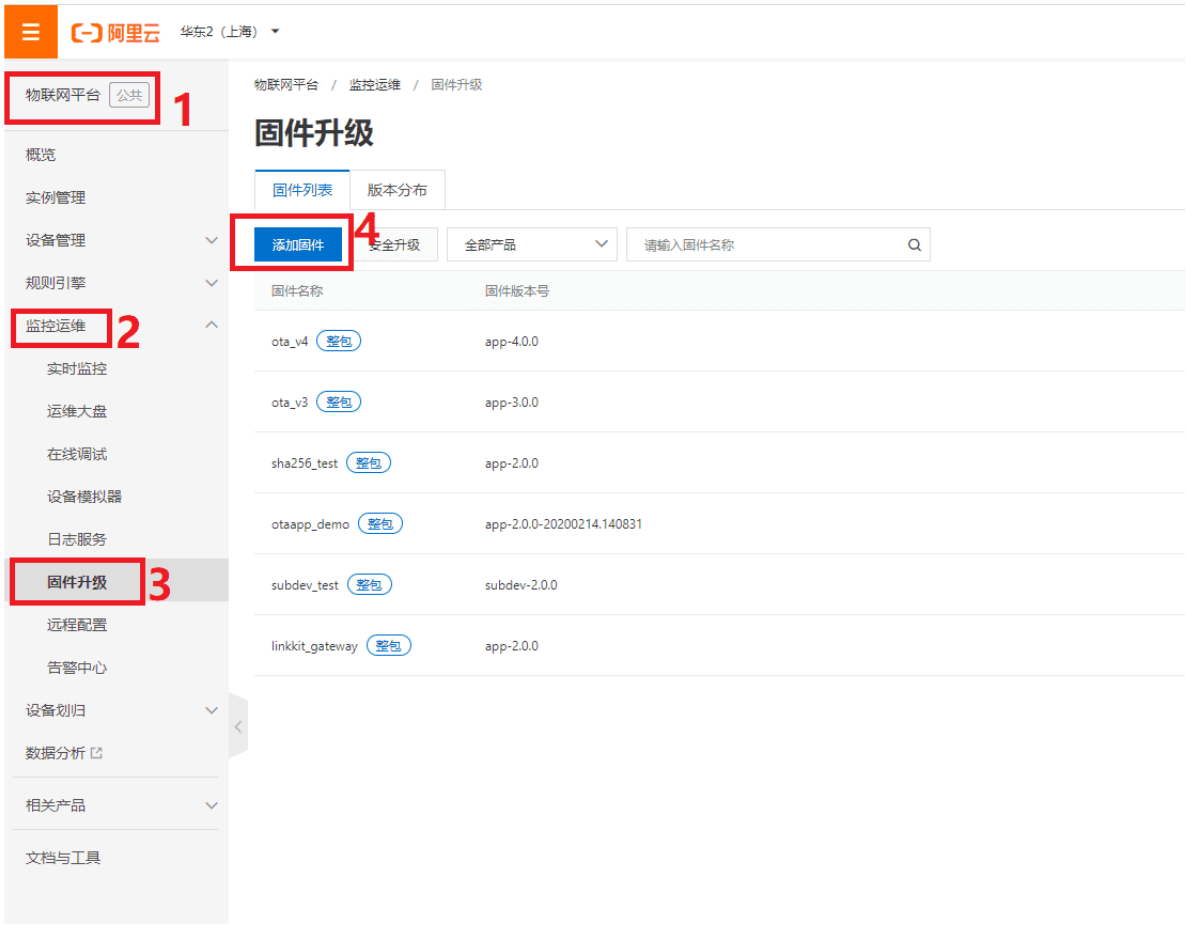
```
[ ]     Download Config                 # 默认采用初始下载参数，选中可配置下载参数
[ ]     RSA Verify Support              # 默认不支持固件验签，选中支持安全验签功能
[ ]     Https Download Support          # 默认支持http下载，选中此项将支持https下载模式
(1024) write flash cache size(bytes) # OTA写flash缓存大小默认是1k，支持自定义
[ ] OTA via uAgent                      # 默认不支持uAgent模式升级，选中支持uAgent方式升级
```

完成配置后保存退出

- **4.编译本地烧录固件**

编译命令：aos make 编译完成后，生成的固件在 `platform/mcu/haas1000/release/write_flash_gui/ota_bin` 目录下，根据前面的烧录文档，先将固件烧录到HaaS 100板子上，重启板子，打开串口终端，配置串口波特率为：**1500000**，连接终端，如果第一次使用板子，需要配置设备的4元组，在终端输入：`identity set pk ps dn ds`，输入成功后以后不用再输入 ,然后配网：在终端输入：`netmgr connect wifi_ssid wifi_password` 连网成功后，登录物联网平台可以看到对应的设备在线；

- **5.编译上云固件及云端操作** 本地烧录完成后，需要做一个高版本固件上传到云端，通过云端操作完成固件的升级，所以需要按照步骤2，修改固件版本号，再按照步骤4编译固件，生成高版本的固件，然后登录物联网平台平台，按如下图顺序操作：



点击新增固件后，如下图将 `platform/mcu/haas1000/release/write_flash_gui/ota_bin/ota_bin/ota_rtos_ota.bin` 上传到云端：

点击确定后，选择验证固件即可开始固件升级；升级结果可以点击“查看”获取详细结果；物联网平台的OTA操作可参考文档阿里云物联网平台固件升级文档

> *注：在做OTA之前确保设备端已连接云端*

6.2. HaaS100文件系统方案介绍

概述

随着智能设备和物联网场景的发展，嵌入式系统中也越来越多地使用到文件系统，如穿戴类设备对传感数据的存储、智能语音设备对图片和语音文件的存储、物联网网关设备对子设备信息的存储等。

AliOS Things自发布至今，已发展到现在的3.x版本，功能日臻完善。其中的文件系统方案，也已经在多个量产项目中规模化使用，功能和特性日渐丰富和成熟。

下图是AliOS Things文件系统的功能特性概要。

架构介绍

AliOS Things中，文件系统层次架构从上到下依次为：应用层、C库层、文件系统层、设备访问层、硬件层。

- **应用层**

  ：用户应用。

- **C库**

  ：libc实现，提供POSIX标准接口服务。

- **VFS**

  ：虚拟文件系统（Virtual File System），可以集成不同的底层文件系统模块。VFS层屏蔽底层文件系统的细节，提供统一的VFS接口服务。

- **文件系统**

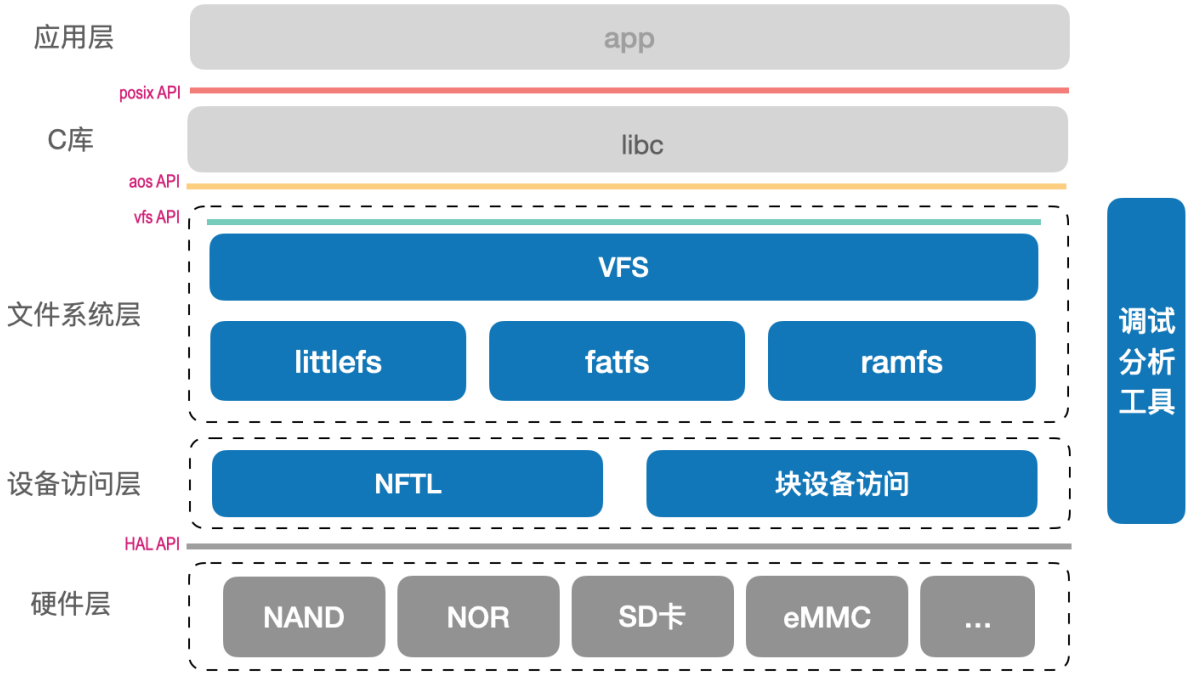
  ：AliOS Things中提供了littlefs、fatfs、ramfs等文件系统功能。

- **设备访问层**

  ：抽象块设备访问逻辑，其中NFTL主要用于NAND Flash设备访问。

- **硬件层**

  ：支持的存储介质包括Nor Flash、NAND Flash、SD卡、eMMC、USB等。



文件系统方案

AliOS Things提供了littlefs、fatfs、ramfs等文件系统模块。其中，littlefs主要用于裸Flash存储介质，fatfs主要用于SD卡、eMMC、USB等存储介质，ramfs用于内存文件系统。

上述文件系统模块，通过VFS层提供统一的文件系统服务能力。不同类型的文件系统，挂载在不同的位置。其中，ramfs默认挂载路径为/RAMFS，littlefs默认挂载点为/data和/system分区，sd卡fatfs默认挂载在/sdcard分区，usb fatfs默认挂载在/usb分区。用户页可以根据需要，修改使用其他挂载路径。

内存文件系统

提供临时文件存储功能，所有的文件系统信息存储在内存中。系统掉电后，所有文件信息丢失；系统重启后，内存文件系统分区重新初始化和挂载。

内存文件系统挂载路径点为 `/RAMFS` ，可以通过 `/RAMFS/<dir>/<file>` 路径访问内存文件系统中的文件和目录。具体的使用方法，请参考示例 `application/example/vfs_demo` 。

FatFS文件系统

FatFs是一种通用的文件系统，在嵌入式系统中实现FAT文件系统。

AliOS Things中也集成了FatFs文件系统，主要用于SD卡、eMMC、USB等存储介质上。其中，使用SD、eMMC的FatFs文件系统默认挂载路径为 `/sdcard` ，使用USB介质的Fatfs文件系统默认挂载在 `/usb` 路径下。

关于在AliOS Things中，如何使用FatFs存储和访问文件，请参考示例：

components/fatfs/example/fatfs_example.c。

littlefs文件系统

为了更好地支持裸Flash场景，AliOS Things集成优化了littlefs文件系统模块。littlefs是一种高度完整的嵌入式文件系统，具有footprint小、抗掉电、磨损平衡等特点，可以更好地支持Flash存储介质的场景。

AliOS Things中，可以支持分区挂载路径为 `/data`。关于如何在AliOS Things上使用littlefs文件系统存储和访问文件，请参考示例应用代码： `components/littlefs/example/littlefs_example.c` 。

NAND Flash文件系统

littlefs在NAND Flash上的问题

上一章节介绍了littlefs文件系统可以适用裸Flash存储介质。对于NAND Flash的情况，虽然littlefs在功能和运行上可行，但是存在以下问题：

- **坏块处理**

  ：由于NAND Flash允许出厂坏块的存在，所以在NAND上处理坏块需要更加健壮的逻辑。littlefs虽然具备完备的运行时坏块处理逻辑，但是对于出厂坏块的处理还需要增强。

- **空间效率问题**

  ：对于NAND Flash，一般物理可擦除块都比较大（如128KB、256KB等）。而littlefs以块存储文件的设计，对于NAND这种场景，大多数情况下存在空间利用率低的问题。例如，对于块大小为256KB的NAND，当系统中存在大量10KB的文件时，整个文件系统的空间利用率只有40%，甚至更低。

为了解决NAND Flash场景下littlefs的上述问题，AliOS Things提供了NFTL（NAND Flash Translation Layer）模块。NFTL模块提供逻辑块到物理块/页的映射机制，通过这种映射机制到达以下目的：

- **坏块屏蔽**

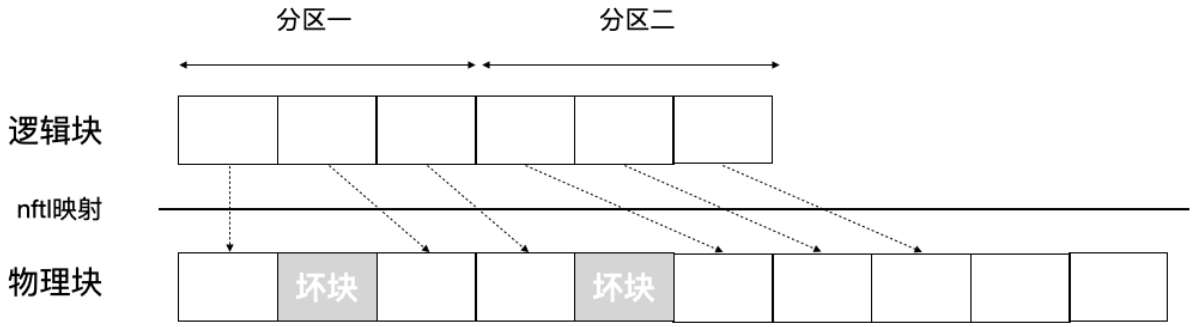
  ：通过逻辑块映射机制，使上层（NFTL的上层，即文件系统或其他使用NFTL的模块）使用的存储块变成逻辑块，上层不必关系逻辑块zai在物理介质上的存储和分布，且不用考虑坏块的问题。因此，在使用NFTL之后，上层可见的存储块是连续的、无坏块状态。

- **大块切分**

  ：通过逻辑块映射机制，使上层可见的逻辑块可以灵活设置大小，而不必和物理块大小一一对应。一般，逻辑块大小可设置成NAND Flash物理页的倍数，如一个逻辑块对应4个物理页。目前，AliOS Things提供的NFTL模块支持逻辑块到物理页的映射（即一个逻辑块对应一个物理页）。通过这种方式，使littlefs在NAND Flash上不必使用较大的物理块作为文件存储块单位，而是以页为单位存储文件，大大提供NAND场景下littlefs的空间利用效率。

NFTL的基本原理

为了实现逻辑块的映射，NFTL使用NAND Flash物理页的spare区域存储逻辑块信息，每次数据写入物理页时，将其对应的逻辑块号写入spare区域。在系统启动时，通过读取spare区域的逻辑块信息，即可恢复逻辑块到物理页的映射关系表。上层模块访问逻辑块时，由NFTL层进行映射转换，将数据实际写入/读取到物理页。如下图所示。



为了加速系统启动中扫描spare区域数据建立映射表的流程，NFTL模块使用每个物理块的最后一页作为加速页，用于存储该物理块上所有页的spare区域的冗余数据（加速备份）。在系统启动中，只需要扫描每个物理块的最后一页数据，即可恢复映射表，缩短了启动时扫描的时间。其缺点是占用了一个物理页用于存储加速备份数据，浪费了一定的空间（在一个典型的系统上，每个物理块含64页，因此将消耗约1.5%的空间）。

NFTL还具备以下功能：

- **掉电安全**

  ：所以数据的写入，不会直接覆盖老数据，而是新写入一个版本号更高的逻辑块数据，通过版本号识别和使用有效数据。当写入数据过程中掉电时，通过ECC校验排除写入不完整数据页，并尝试使用低版本号数据页作为有效数据，从而达到掉电保护的目的。

- **坏块管理**

  ：在擦写过程中遇到坏块时，NFTL将自动进行数据迁移，并重新选取一个可用块用于数据操作。对于读取数据时的错误，目前主要依赖ECC校验保证。此外，由于NFTL不存在超级块的概念，因此对于出厂坏块问题的处理更加健壮可靠。

- **磨损平衡**

  ：NFTL选取数据块进行擦写时，使用了一种动态平衡机制来达到磨损平衡的目的。

- **垃圾回收**

  ：适时启动垃圾回收流程，保证垃圾块/页的回收利用。

如果想了解NFTL的具体细节，请参考NFTL的README文件（`core/nftl/README.md`）和代码实现（`core/nftl`）。

NFTL+littlefs方案的移植与使用

目前，HaaS100平台未支持NAND Flash。用户如果使用NAND Flash作为文件系统存储介质，按照以下步骤进行必要的移植适配后，即可运行和使用NFTL+littlefs文件系统功能。

1. 根据您项目的实际情况，划分和修改Flash分区表，请注意划分给NFTL和文件系统使用的分区偏移位置和大小。注意，划分分区时，物理上只有NFTL分区的概念，没有文件系统分区的概念。NFTL分区包含文件系统分区部分和NFTL管理块部分，因此实际划分给文件系统的空间比NFTL分区大小略小。
2. 根据分区表划分，修改nftl配置文件。修改文件路径：

   `core/nftl/inc/nftl_conf.h`

。请注意修改如下配置项：

- **NFTL_PAGE_SIZE**

  ，请配置为NAND Flash的物理页大小。

- **NFTL_PAGE_PER_BLOCK_NUMS**

  ，请配置为NAND Flash物理块上的实际页数。

- **NFTL_LOGICAL_PARTITION_NUM**

  ：配置NFTL逻辑分区的数量，如支持/data分区则配置为1，支持/data、/system（A备份）、/system（B备份）则配置为3.

- **NFTL_PHYSICAL_PARTITION0/1/2**

  ，NFTL对应的分区号，注意与实际Flash分区操作对应。

- **NFTL_PART0/1/2_PHY_BLOCK_NUMS**

  ，NFTL分区所占用的物理块数量，请注意与分区表对应。

- **NFTL_PART0/1/2_RAW_BLOCK_NUMS**

  ，NFTL分区内文件系统所占用的物理块数量。

  建议文件系统所占空间（物理块数）不超过其所在的NFTL分区空间的98%

  。因为，NFTL要预留一定的物理块用于坏块备份和管理。

3. 修改和配置littlefs文件系统分区的大小，配置文件修改路径为：

   `components/fs/littlefs/platform_conf/<platform>/platform_conf.h`

   。

参考如下示例和说明：

```
/* 配置分区数量，第一个分区为/data，第二个为/system */
#define LITTLEFS_CNT 2
/* 配置为NAND Flash的物理页大小 */
#define PROG_SIZE 4096
/* 配置块大小，目前仅支持配置为1 */
#define PAGE_NUMS_ON_BLOCK 1
/* /data分区所占用的block数量，注意与与NFTL的RAW_BLOCK_NUMS匹配。 */
#define DATA_BLOCK_NUMS 1024
/* /system分区所占block数，仅在LITTLEFS_CNT配置为2时有效。注意与与NFTL的RAW_BLOCK_NUMS匹配。 */
#define SYSTEM_BLOCK_NUMS 512
```

4. Flash HAL接口的对接实现：

- **通用接口实现**

  ：原型声明见文件

  `include/aos/hal/flash.h`

  ，可以参考HaaS100平台提供的实现

  `platform/mcu/haas1000/hal/flash.c`

。

- **NAND特有接口实现**

  ：原型声明位于

  ```
  include/aos/hal/nand_flash.h
  ```

  。

5. 镜像编译、烧录和运行，请参考具体平台的烧录步骤。

总结

AliOS Things提供了丰富的文件系统方案，以及多样化的存储介质支持，特别是对NAND Flash的完备支持。同时提供了基于POSIX标准的API，使基于标准接口开发的上层应用可以很轻易地进行移植。

如果您对AliOS Things的文件系统方案感兴趣，欢迎访问AliOS Things Github主页。如果您有任何建议或疑问，欢迎通过钉钉群或者阿里云开发者社区与我们联系。

6.3. 基于HaaS 100搭建智能家居应用

本文详细介绍如何基于HaaS平台快速接入阿里云IoT，搭建智能生活应用场景。

1.设备端环境搭建

参考HaaS快速开始。

2.云端配置

登录智能生活物联平台(https://living.aliyun.com/)。未注册阿里云账户的用户，请先完成账户注册。

2.1 创建项目

如下图所示，点击右上角创建新项目。

2.2 创建产品

项目创建完成后，创建新产品。

2.2.1 产品类型选择

如下图，产品创建时信息填写说明：

1.产品名称，按您喜好填就行；

2.所属品类，智能生活平台提供覆盖行业内大部分的产品种类的物模型，我们这里使用HaaS开发板模拟智能灯，所以选择“电工照明/灯”；

3.节点类型，直连设备，选“设备”；

4.直连接入，选非网关接入；

5.我们使用HaaS Wi-Fi网络接入，连网方式选择“WiFi”；

6.数据格式，选择“ICA”即Alink Json格式，也可以选择raw数据格式（设备端不理解具体用户协议，需在云端平台设置js脚本将二进制协议转换称ICA协议）；

7.设备端默认不支持ID2认证，选“否”。

2.2.2 完善产品信息

如下图所示：

1.勾选使用公版APP控制产品。

2.完善左侧惊叹号提示的配置设置，对于已经有默认数值的按默认确认即可。

3.其中需要注意的有两个地方：

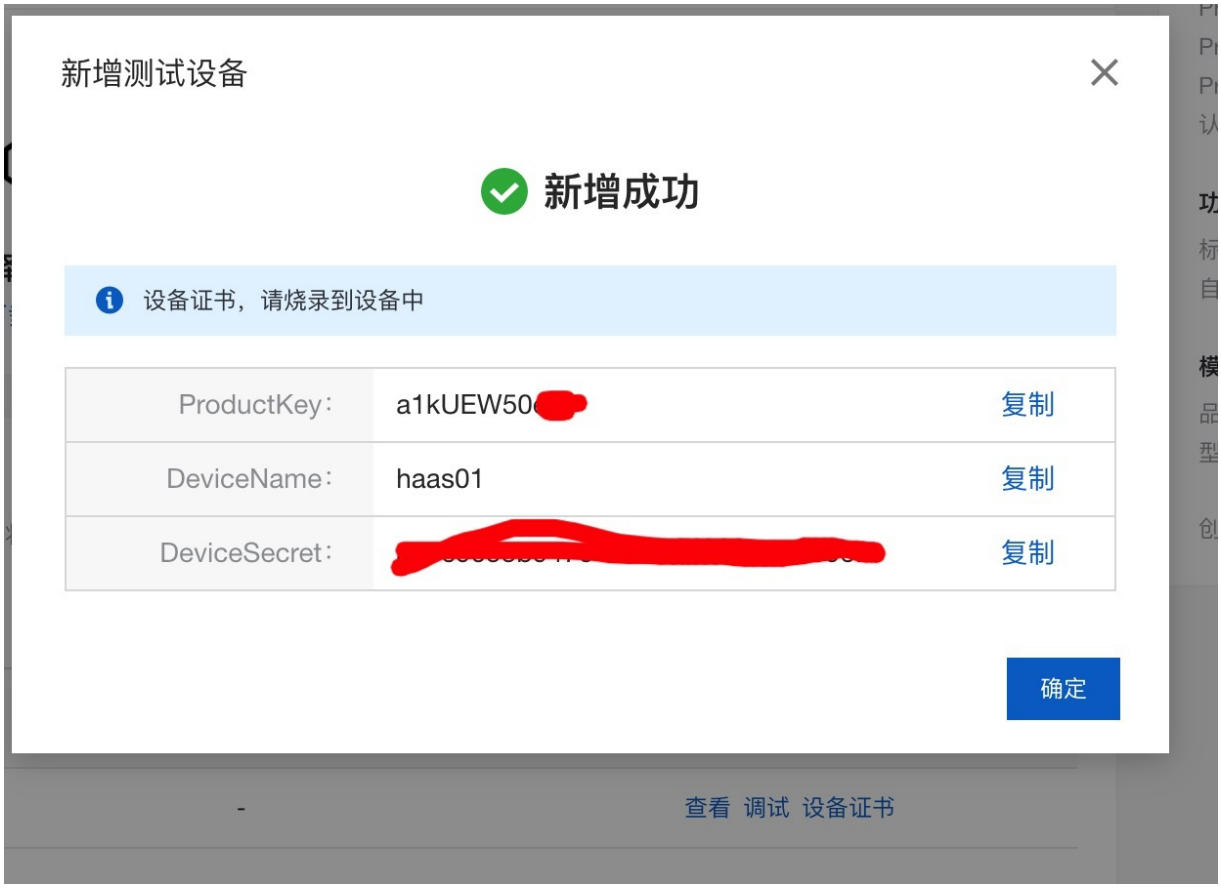
1)配网引导页面点自定义配网设置。

2)多语言版本设置时对于未设置的数值都填一个默认值，如：

一	亮度
百分比	100%
摇一摇模式	0
作业模式	0
音乐流水模式	0
音乐能量模式	0
音乐律动模式	0
复古disco模式	0
浪漫时刻模式	0
星月相伴模式	0
活力无限模式	0

4.模组类型选择未认证。

5.创建一个设备，并拷贝其三元组(ProductKey/DeviceName/DeviceSecret)信息。



6.同时拷贝产品密钥(ProductSecret)，用于配置设备端配网：


HaaS智能控制
更新时间：2020-09-16 21:03:54
基本信息 编辑
所属分类：灯
节点类型：设备
通讯方式：WIFI
数据格式：ICA标准数据格式（推荐）
Product Key：a1kUEW50es1
Product Secret：
复制
隐藏
Product Id：5922049
认证方式：设备密钥
功能定义 查看
标准功能：9
自定义功能：0
模组 重选
品牌：上海庆科
型号：EMW3072 详情
创建时间：2020-09-16

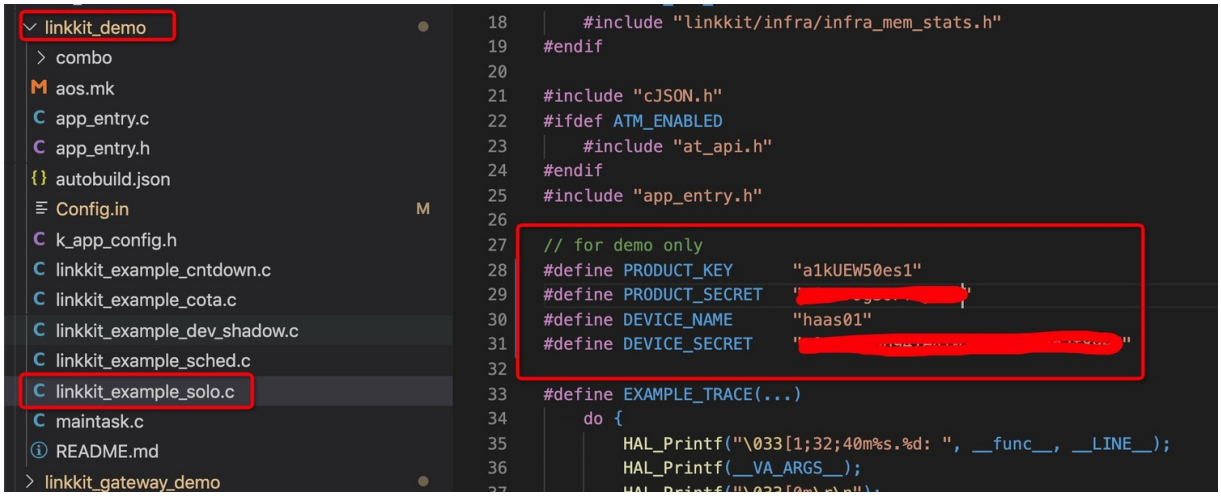
7.完成开发：

8.安装手机端APP,如图扫描二维码下载云智能APP:

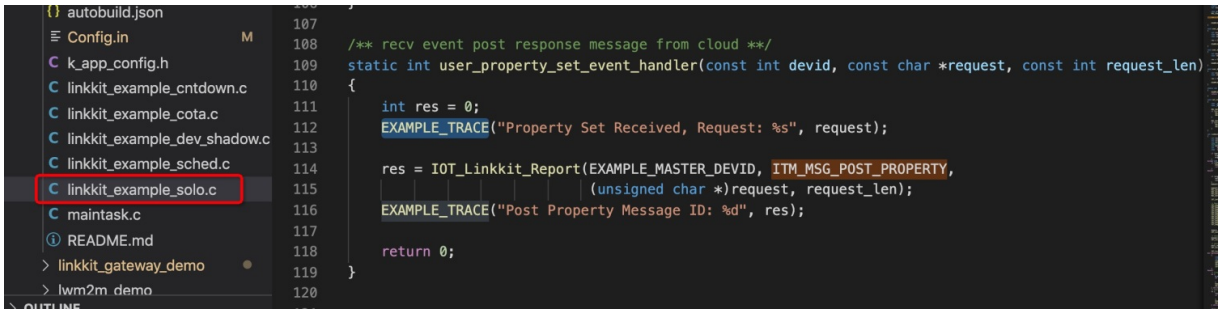
至此，云端配置完成。

3 设备端配置

1.修改端上代码 application/example/linkkit_demo/linkkit_example_solo.c 中三元组信息，如下图，其中三元组（ProductKey/DeviceName/DeviceSecret,加上ProductSecret后也可以称为四元组）信息从2.2.2章节获取。



2.修改属性设置回调函数，可在此函数中加上自己的代码，如根据云端属性控制IO操作等：



更多教程可参考Linkkit使用文档。

3.编译

```
aos make linkkit_demo@haas100 -c config
aos make
```

4.按章节1指导方式烧入固件。

4 端云联调

4.1 配网与绑定

4.1.1 设备配网

1.通过命令行配网，串口输入如下指令可完成配网：

1. 输入"netmgr -t wifi -b 1" 打开连网成功后会自动保存AP信息的功能。
2. 输入"netmgr -t wifi -c {ssid} {password}" 连接名为ssid的路由器AP。
3. 连网成功后，固件会自动保存当前的AP信息。
4. 重启后，固件会自动会连上次连接成功并保存的AP。
5. 如需切换新的AP连接，请重新输入命令"netmgr -t wifi -b 1"和"netmgr -t wifi -c {ssid} {password}"，新AP连接成功后自动覆盖旧的AP信息并保存。

注意： {ssid} {password}替换为自己的路由器配网信息，如netmgr -t wifi -c my_wifi 12345678。

关键日志：

```
ssid=aos_test_01
ip_address=192.168.18.109
address= 0:80:3d:64: 8:77
wpa_state=COMPLETED
```

获取到IP地址后程序会立即进行linkkit连云；连上云关键日志：

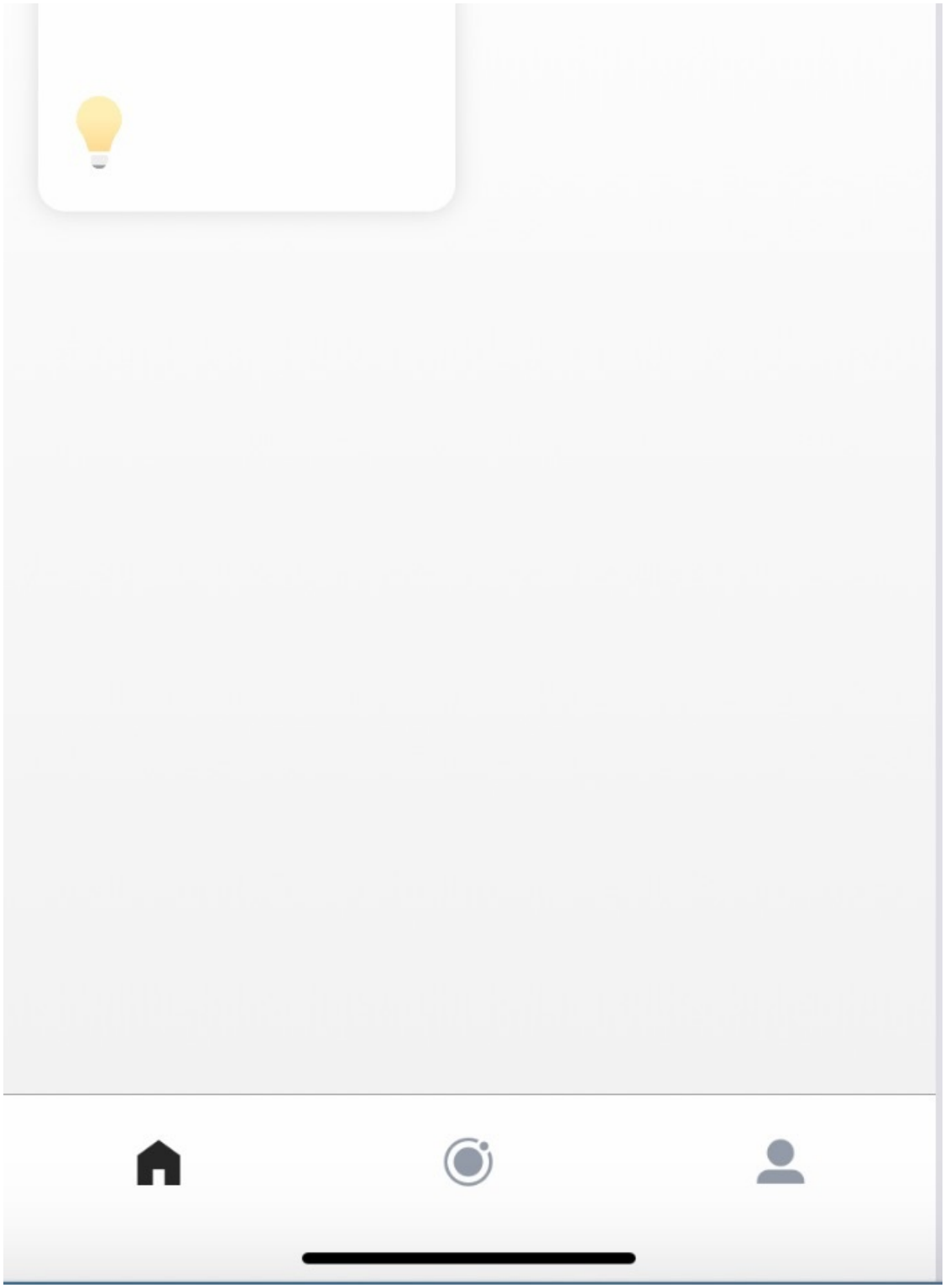
```
[Jan 01 00:00:11.613]<I>HAL_TLS  ok
[Jan 01 00:00:11.613]<I>HAL_TLS   . Setting up the SSL/TLS structure...
[Jan 01 00:00:11.613]<I>HAL_TLS  ok
[Jan 01 00:00:11.614]<I>HAL_TLS Performing the SSL/TLS handshake...
[Jan 01 00:00:11.731]<I>HAL_TLS  ok
[Jan 01 00:00:11.731]<I>HAL_TLS   . Verifying peer X.509 certificate..
[Jan 01 00:00:11.731]<I>HAL_TLS certificate verification result: 0x00
[Jan 01 00:00:11.731]<I>MQTT connect params: MQTTVersion=4, clientID=a1Oi7Q7h3LV.haas_01|ti
mestamp=2524608000000,_v=sdk-c-3.0.1,securemode=2,signmethoV
[Jan 01 00:00:11.772]<I>MQTT mqtt connect success!
```

至此设备已经完成配网并使用我们自己创建的三元组（product key/device name/device secret）连接到智能生活平台。

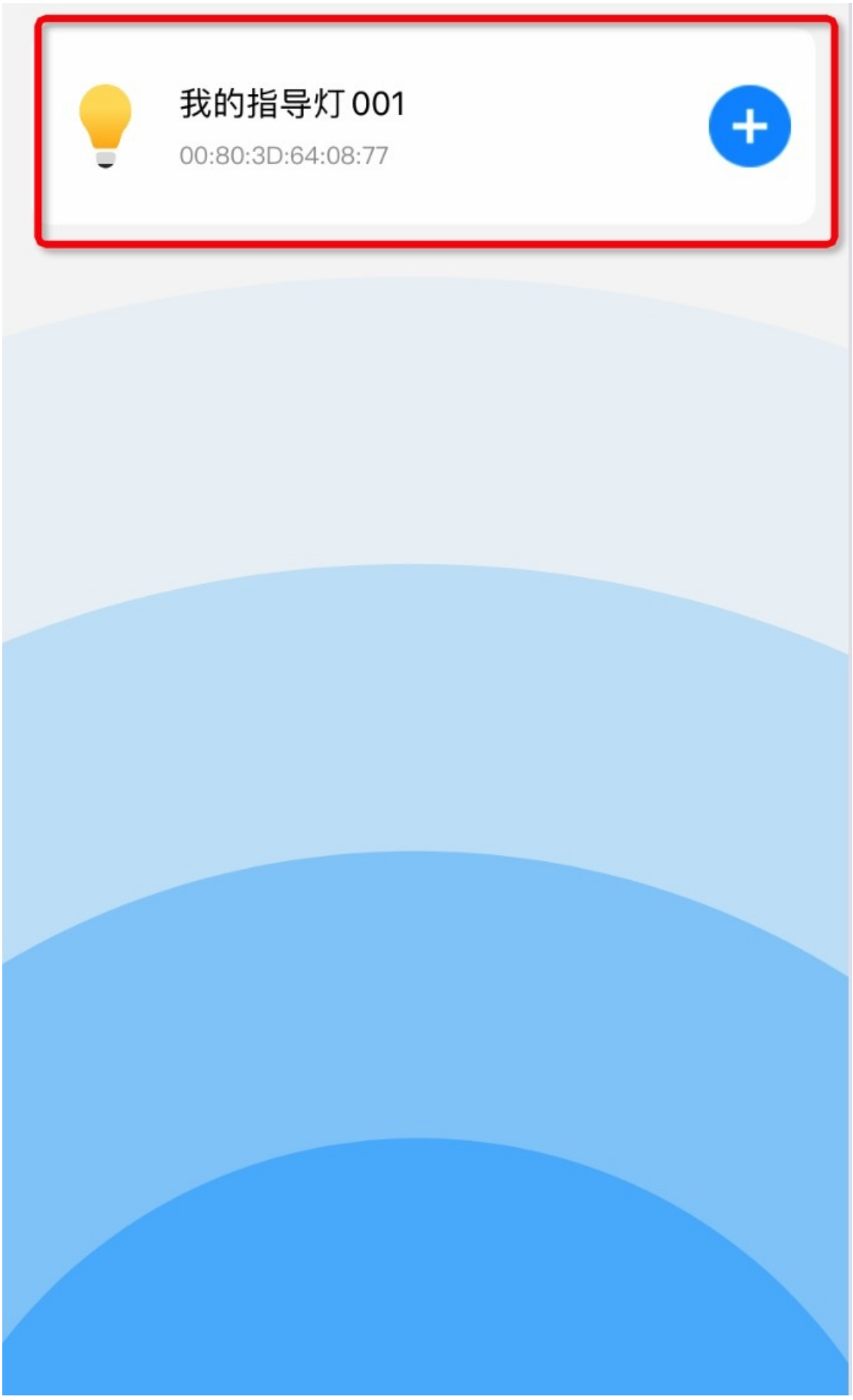
注：HaaS100开发板APP配网功能还在开发中，请持续关注代码更新。

4.1.2 设备与云智能APP绑定

打开步骤2.2安装的云智能APP,保证手机更设备连接同一个APP.点击右上角红色标注"+" 按钮，开始设备查找。



约数秒钟后找到设备，如下图，点击完成。


我的指导灯001
00:80:3D:64:08:77

至此绑定完成。

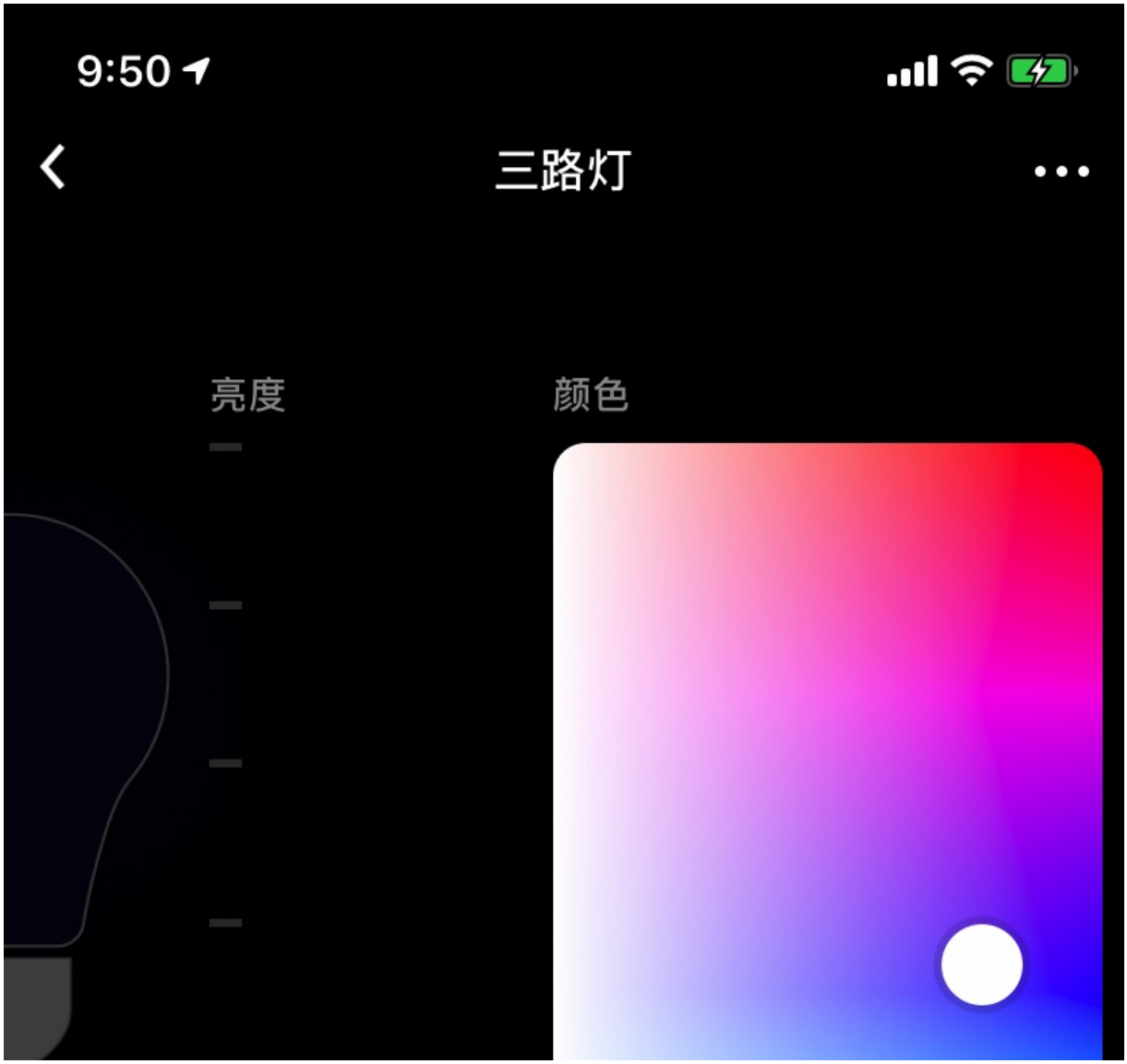
如果此处找不到设备,请检查:

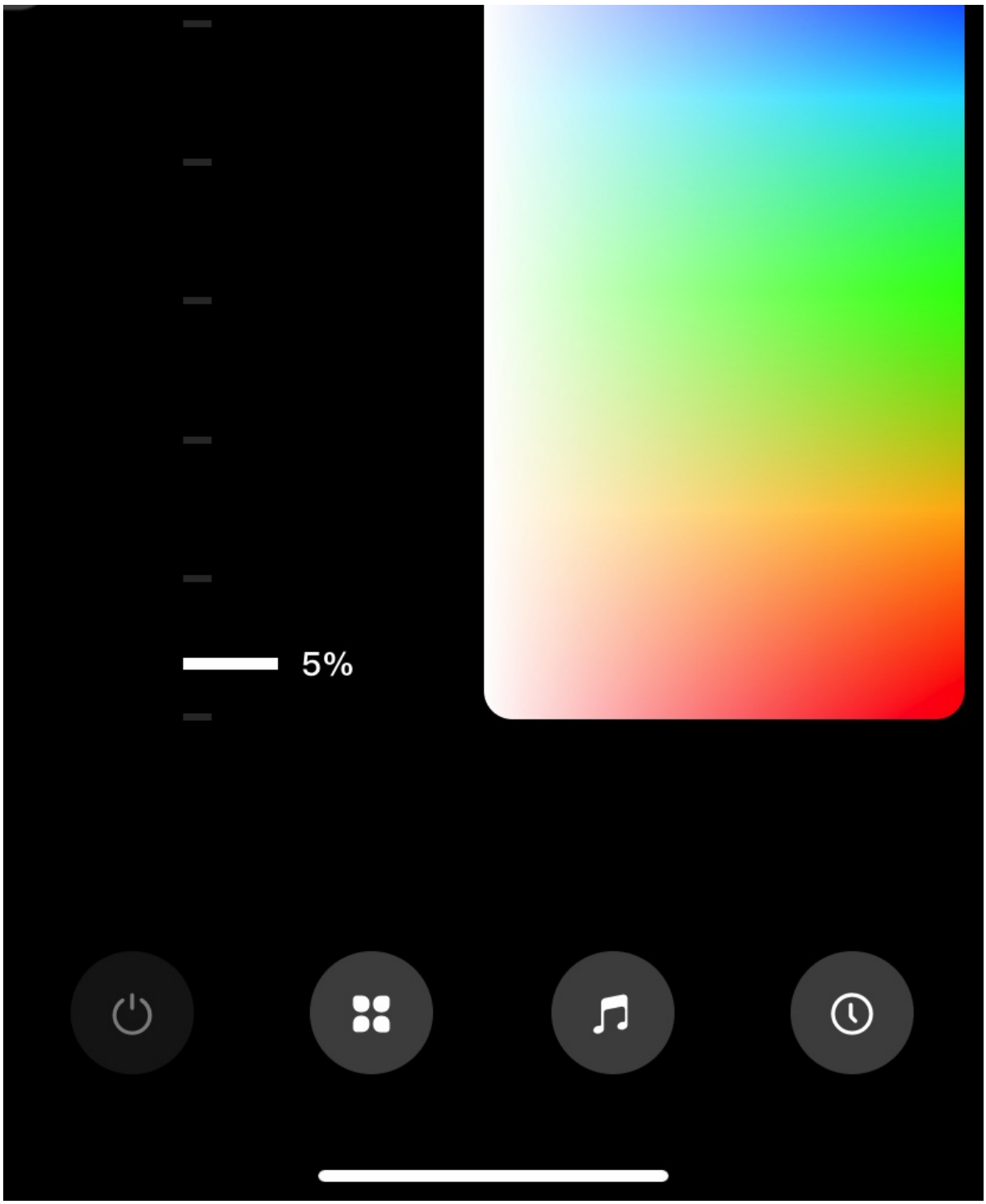
1. 创建的设备是否处于在线状态;

2. 2.2.2章节第1和第7步骤是否完成。

4.2 控制测试

设备绑定完成后，进入如下页面，可以通过点击图中不同模块进行命令下发。同时关注设备端日志打印。





典型日志：

```
[Jan 01 00:02:12.197]<I>MQTT Downstream Topic: '/sys/a1Oi7Q7h3LV/haas_01/thing/service/property/set'
[Jan 01 00:02:12.197]<I>MQTT Downstream Payload:

< {
<     "method":"thing.service.property.set",
<     "id":"575303451",
<     "params":{
<         "HSVColor":{
<             "Saturation":84,
<             "Value":5,
<             "Hue":232
<         }
<     },
<     "version":"1.0.0"
< }

[Jan 01 00:02:12.198]<I>DM thing/service/property/set
[Jan 01 00:02:12.198]<I>DM Send URI: /sys/a1Oi7Q7h3LV/haas_01/thing/service/property/set_reply, Payload: {"id":"575303451","code":200,"data":{}}
[Jan 01 00:02:12.200]<I>MQTT Upstream Topic: '/sys/a1Oi7Q7h3LV/haas_01/thing/service/property/set_reply'
[Jan 01 00:02:12.200]<I>MQTT Upstream Payload:

> {
>     "id":"575303451",
>     "code":200,
>     "data":{
>     }
> }
```

当然你还可以在属性或服务下发回调函数中添加自己的代码完成更多的功能，比如控制IO口操作真实的外设。