



OPEN TOGETHER

AliOS Things Developer Kit Quick Start

Developer Kit 快速入门

Revision 1.01

2018 年 6 月 25 日

目 录

目 录	- 2 -
1. 概述	- 4 -
1.1 开发平台	- 4 -
1.2 出厂软件和 SDK	- 4 -
2. 开始开发	- 5 -
2.1 环境搭建	- 5 -
2.1.1 驱动程序	- 5 -
2.1.2 ST-LINK Utility	- 5 -
2.1.3 Python2	- 5 -
2.1.4 pip	- 6 -
2.1.5 aos	- 6 -
2.1.6 AliOS Things Studio	- 7 -
2.1.7 git	- 7 -
2.2 代码简介	- 7 -
2.3 开始编译	- 9 -
2.4 开始烧写	- 9 -
2.4.1 虚拟 U 盘烧写	- 9 -
2.4.2 ST-LINK Utility 烧写	- 9 -
2.4.3 AliOS Things Studio 烧写	- 10 -
2.5 开始调试	- 10 -
3. 例程介绍	- 13 -
3.1 udataapp	- 13 -
3.2 cameraapp	- 15 -
3.3 其他例程陆续完善中	- 16 -

修订记录

版本	日期	作者	变更描述
0.9	2018.05.29	Notion	初始版本
1.0	2018.06.06	Notion	添加了 linkkitapp 例程介绍
1.01	2018.06.25	Notion	添加了代码简介，勘误完善例程介绍

1. 概述

1.1 开发平台

Developer Kit 是一款由上海诺行信息技术有限公司基于 STM32L496VGx 设计的高性能物联网开发板。它的主要软件平台为开源的 AliOS Things，开发者可以基于此快速的开发出各种物联网设备与产品。

AliOS Things 是面向 IoT 领域的轻量级物联网嵌入式操作系统。致力于搭建云端一体化 IoT 基础设备。具备极致性能，极简开发、云端一体、丰富组件、安全防护等关键能力，并支持终端设备连接到阿里云 Link，可广泛应用在智能家居、智慧城市、新出行等领域。

1.2 出厂软件和 SDK

Developer Kit 的软件分成了出厂版本和 SDK。

- 出厂版本：提供给工厂生产人员使用，用来检验测试出厂硬件各个外设器件是否能够工作正常。工具包套件中仅包含了编译好的二进制文件，开发者也可以用来对单板进行测试或者观察外设的功能特性。相关使用方法请参考文档《AliOS Things Developer Kit 用户操作指导》。
- SDK：提供给物联网开发者与高校教育培训机构使用，用来快速评估与开发出各种物联网产品。SDK 内容包括编译烧写相关的工具和主要软件平台 AliOS Things。Developer Kit 板级代码已经加入 AliOS Things 源码树中，以后也会不断地进行完善与更新。

2. 开始开发

2.1 环境搭建

开始开发 AliOS Things 之前，需要安装一系列的工具软件。本章节主要介绍 Windows 环境下开发环境的搭建过程。

2.1.1 驱动程序

打开软件包，解压 en.stsw-link009.rar 到当前目录。

将附带的 USB 线缆一头插入 Developer Kit 的 USB ST-Link 口，另外一头连接 PC。打开设备管理器，将多出来的未知设备更新驱动程序，选中刚才解压的目录路径，完成安装

2.1.2 ST-LINK Utility

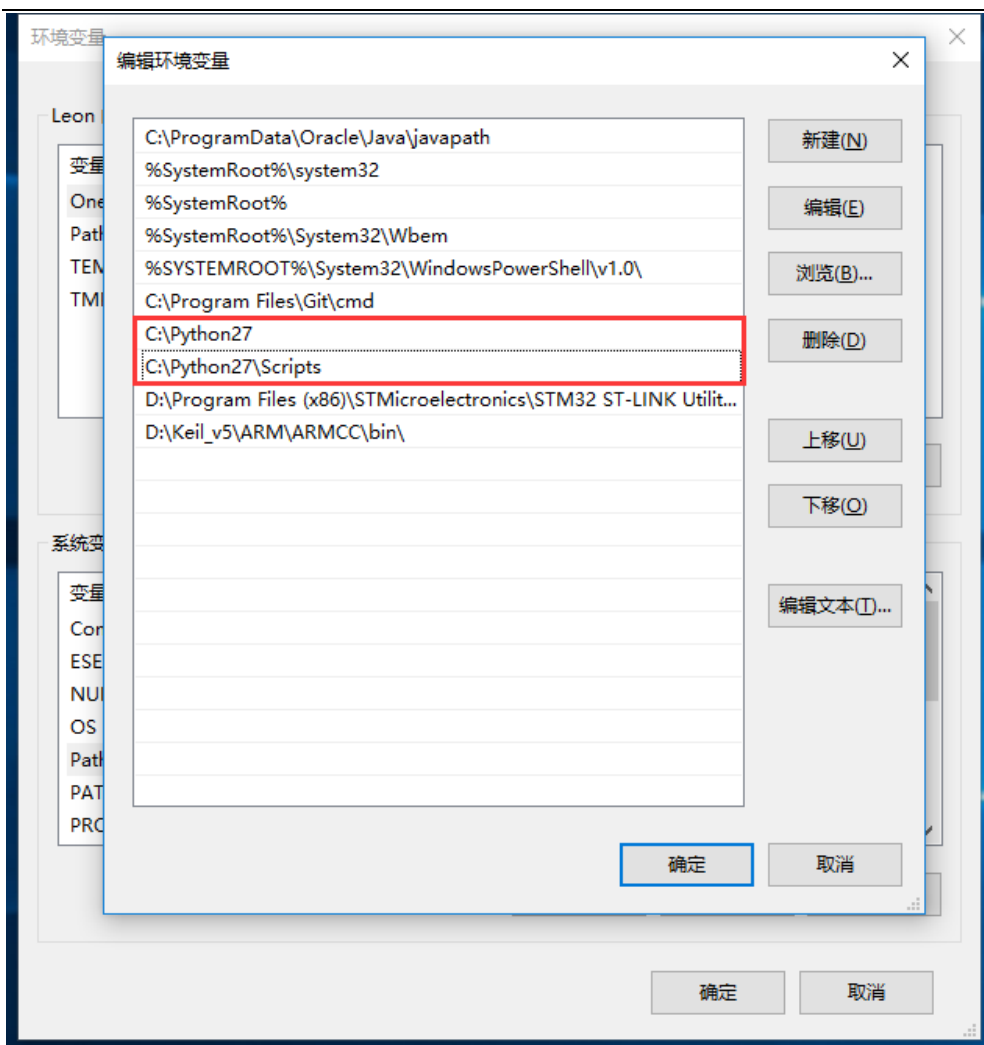
打开软件包，解压 en.stsw-link004.zip 到当前目录。打开解压后的安装程序一路完成安装。

2.1.3 Python2

打开 Python 官网，选择相应的 2.x 版本（当前最新版本为 Python 2.7.15）下载并安装：

<https://www.python.org/downloads/windows/>

安装完成后，在系统属性中为的系统环境变量 PATH 加入相应的路径（根据自己的安装目录）：



2.1.4 pip

打开 pip 官方网站, 选择最新的压缩包版本(当前最新版本为 pip-10.0.1.tar.gz) 下载:

<https://pypi.org/project/pip/#files>

完成后解压到任意目录, 打开命令提示符, 进入到刚才解压到的根目录中, 输入命令安装 pip:

```
python setup.py install
```

2.1.5 aos

Aos 是阿里的代码管理编译工具。在安装 aos 之前, 需要安装或更新相关依赖软件。在刚才的命令提示符中输入如下命令:

```
$ pip install --upgrade setuptools
```

```
$ pip install --upgrade wheel
```

完成后输入如下命令安装 aos:

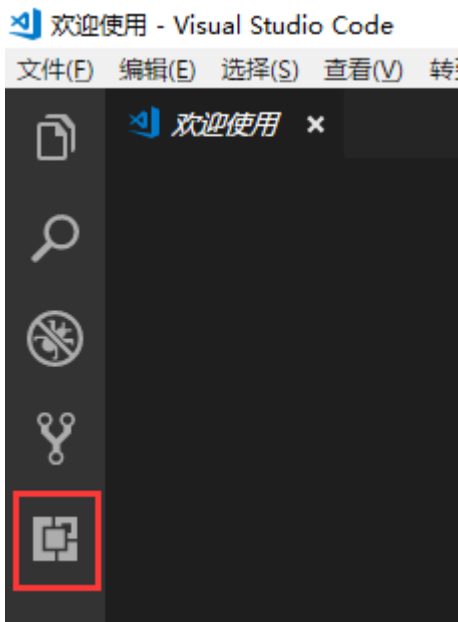
```
$ pip install --upgrade aos-cube
```

2.1.6 AliOS Things Studio

AliOS Things Studio 是阿里基于 Visual Studio Code 的一套图形化 IDE。进入官网下载并安装:

<https://code.visualstudio.com/>

完成后打开工具, 在最左边的工具条中选择点击扩展, 分别安装 C/C++ 插件和 alios-studio 插件:



2.1.7 git

为了同步最新的 AliOS Things 代码, 需要安装 git。进入官网下载并安装:

<https://git-scm.com/downloads>

完成后使用 Git GUI 或者 Git Bash 将最新代码克隆到本地:

```
git clone https://github.com/alibaba/AliOS-Things.git
```

```
git checkout master
```

2.2 代码简介

AliOS Things 目前处于版本快速迭代期, 你看到的目录结构有可能跟本章节介

绍的稍微不一样。这里只简单介绍一下比较重要的地方。

AliOS-Things



└─.vscode	IDE 编译调试配置文件
└─board	工程目标板和板级代码对应文件夹
└─developerkit	对应工程 xxx@developerkit
...	
└─aos	板级总线初始化、分区、核心模块驱动源码
└─camera_hal	soc camera 组件
└─hal	AliOS Hal 适配头文件
└─inc	板级代码头文件
└─irda_hal	单板红外组件
└─Src	ST CUBE MX 自动生成代码
└─starterkit	对应工程 xxx@starterkit
...	
...	
...	
└─build	aos 脚本和交叉编译工具链
└─device	AliOS 标准驱动程序
└─example	工程应用代码对应文件夹
└─cameraapp	对应工程 cameraapp@xxx, developerkit 拍照存 sd 卡 demo
└─developerkitgui	对应工程 developerkitgui@xxx, developerkit GUI demo
└─helloworld	对应工程 helloworld@xxx, 所有单板 helloworld 打印 demo
└─linkkitapp	对应工程 linkkitapp@xxx, 所有单板 linkkit 入云 demo
└─mqttapp	对应工程 mqttapp@xxx, 所有单板 mqtt 入云 demo
└─starterkitgui	对应工程 starterkitgui@xxx, starterkit GUI demo
└─uDataapp	对应工程 uDataapp@xxx, 所有单板 linkkit + sensor 数据入云 demo
...	
└─framework	AliOS 子系统核心代码
└─include	AliOS 系统调用公共头文件
└─kernel	AliOS 内核核心代码
└─out	aos 编译结果生成目录
└─platform	平台级代码
└─mcu	
└─stm32l4xx_cube	starterkit 和 developerkit 所属平台级代码
└─aos	平台级初始化代码

		└─Drivers	平台级总线驱动代码
		└─hal	AliOS Hal 适配源文件
		...	
		└─projects	legacy 工程文件，目前已经切换到 aos 加 vscode 环境
		...	

AliOS Things 代码的工程名格式为 “应用名字@目标板名字”。应用名字对应为 example 目录下对应的文件夹名；目标板名字对应为 board 目录下对应的文件夹名。例如 Developer Kit 的 helloworld 例程工程名为 helloworld@developerkit。

2.3 开始编译

打开 AliOS Things Studio, 点击左上角菜单栏上的文件->打开文件夹, 选择 AliOS Things 代码根目录。

打开后点击工具左下角蓝色框条中的  图标, 先输入应用名字, 敲回车, 再输入目标板名字, 敲回车。然后点击旁边的  图标开始编译。第一次编译有可能报错, 再点击一次一般就能够成功了。编译成功后的二进制文件放在 out\helloworld@developerkit\binary 里面。

2.4 开始烧写

Developer Kit 提供了三种烧写方式, 分别为虚拟 U 盘烧写、ST-LINK Utility 烧写和 AliOS Things Studio 烧写。使用方法如下 (推荐使用虚拟 U 盘烧写方式):

烧写和调试之前需要使用附带的 USB 线缆一头插入 Developer Kit 的 USB ST-Link 口, 另外一头连接 PC。此时在 PC 设备管理器中可以看到会自动生成一个虚拟 UART 口。这个 UART 是单板的软件调试口, 系统运行后可以观察到调试输出信息。


2.4.1 虚拟 U 盘烧写

将单板 USB ST-Link 连接到 PC, 此时资源管理器会自动装载一个虚拟的 USB 存储器, 直接将编译好的 binary 拷贝到这个 USB 存储器, 即可自动烧写。

烧写完成后单板会自动 reset, 并且可以观察到虚拟的 USB 存储器被重新装载。


2.4.2 ST-LINK Utility 烧写

打开 ST-LINK Utility, 点击工具栏中的  图标连接设备, 连接成功后, 点击 

打开 binary，然后点击  打开烧写界面，最后点击 start 开始烧写。

烧写完成后单板会自动 reset，可以观察到调试 UART 口输出信息。

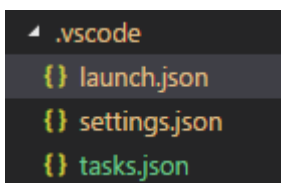
2.4.3 AliOS Things Studio 烧写

AliOS Things Studio 在最后一次编译完成后，直接点击左下角蓝色框条中的  图标进行烧写。注意这种方法烧写单板后，可能不会自动 reset。如果出现烧写后没有运行或者运行异常，尝试拔插 USB 线缆即可。

2.5 开始调试

在开始调试之前，需要先对调试器进行板级配置。

使用 AliOS Things Studio 打开单板代码目录，可以观察到左侧资源视图中生成了一个.vscode 目录，打开此目录下的文件 launch.json。



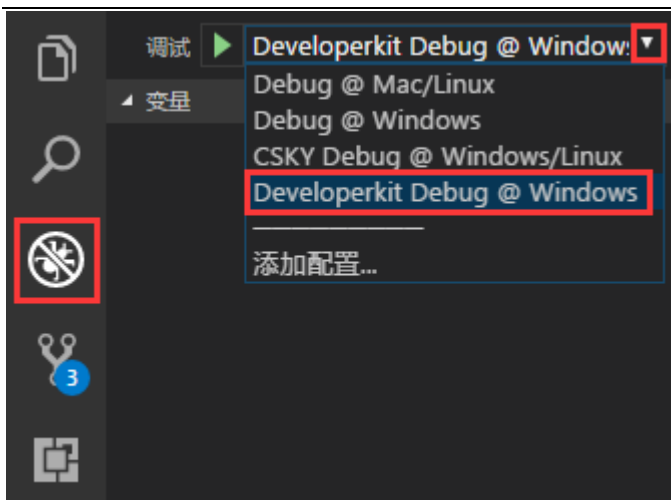
可以看到默认会有几个单板调试配置，开发者可以选择在已经存在配置里面修改，或者创建一组新的配置。

这里以创建新配置为例，将 Debug @ Windows 这一组配置复制下来，添加到最后。修改并适配到目前的工程，主要注意四条配置：name、program、setupCommands 和 miDebuggerPath，相关路径使用斜杠或者双反斜杠。我这里以设置 name 为 Developerkit Debug @ Windows 为例，如下：

```
{
  "name": "Developerkit Debug @ Windows",
  "type": "cppdbg",
  "request": "launch",
  "program":
"${workspaceRoot}/out/helloworld@developerkit/binary/helloworld@developerkit.elf",
  "args": [],
  "stopAtEntry": true,
  "cwd": "${workspaceRoot}",
  "environment": [],
  "externalConsole": true,
  "miDebuggerServerAddress": "localhost:4242",
```

```
"serverLaunchTimeout": 2000,
"targetArchitecture": "ARM",
"setupCommands": [
  {
    "text": "cd ${workspaceRoot}"
  },
  {
    "text": "source .gdbinit"
  },
  {
    "text": "target remote localhost:4242"
  },
  {
    "text": "file out/helloworld@developerkit/binary/helloworld@developerkit.elf"
  },
  {
    "text": "break application_start"
  }
],
"customLaunchSetupCommands": [],
"launchCompleteCommand": "exec-run",
"osx": {
  "MIMode": "gdb",
  "miDebuggerPath": "arm-none-eabi-gdb"
},
"linux": {
  "MIMode": "gdb",
  "miDebuggerPath": "arm-none-eabi-gdb"
},
"windows": {
  "MIMode": "gdb",
  "miDebuggerPath":
"${workspaceRoot}/build/compiler/gcc-arm-none-eabi/Win32/bin/arm-none-eabi-gdb.exe"
}
}
```

调试器配置完成后，在最左侧的工具栏中点击调试图标，并选择调试器配置：



最后，在最左侧的工具栏中点击资源管理器，选择你想要调试的代码段下断点，按下 F5 开始进行在线调试。

3. 例程介绍

3.1 udataapp

Link Develop 是阿里云针对物联网领域提供的端到端一站式开发平台，可覆盖各个物联网行业应用场景，主要解决物联网开发领域开发链路长、技术栈复杂、协同成本高、方案移植困难的问题，提供了从硬件设备、模组、数据、服务 API、Web 应用开发到移动 APP 开发全链路的开发流程、框架 / 引擎和调试工具，并可将成熟的开发产出物对接阿里云云市场进行售卖，为开发者实现商业收益。

udataapp 是一个为开发者展示如何通过 developerkit 接入 Link Develop 平台的例程。该例程会启动相关 sensor 读取环境数据，并将数据自动上传到阿里的智能生活开放平台上。

开始调试 Linkkitapp 之前，需要先进行云端配置，此章节不会对此做详细介绍。云端配置请参考 AliOS-Things 官方 WIKI 相关说明[第二章内容](#)（注意[云端配置完成后](#)再转到此文档）：

<https://github.com/alibaba/AliOS-Things/wiki/%E4%BC%A0%E6%84%9F%E5%99%A8%E6%95%B0%E6%8D%AE%E4%B8%80%E9%94%AE%E5%BC%8F%E4%B8%8A%E4%BA%91%E7%9A%84%E4%BD%BF%E7%94%A8%E8%AF%B4%E6%98%8E>

完成后云端配置后，就要开始进行终端设备的开发了。首先将云端配置中的 TLS 信息转换成 C 语音的 JSON 格式，打开页面：

<https://www.sojson.com/yasuo.html>

将得到的 TLS 信息粘贴进去，并点击“压缩并转义”。

在 AliOS Things Studio 中打开 example\userDataapp\userData_tsl.c，找到全局字符串数组变量 TSL_STRING，将刚刚得到的 JSON 格式的 TLS 字段粘贴进去；打开 framework\protocol\linkkit\iotkit\sdk-encap\imports\iot_import_product.h，找到四元组的定义（udataapp 例程一般是最后一个 else），将其替换成云端中得到的字段，如下：

```
#else
#define PRODUCT_KEY          "a1QD95fzs gb"
#define PRODUCT_SECRET       "L55553DivHnQFLCz"
#define DEVICE_NAME          "2Bbs4oASBELP0RWaENfu"
#define DEVICE_SECRET        "6Ds5oQjo5bE7CtUroxoIBHmlmxsIw83m"
#endif
#endif
```

继续打开 framework\userData\userData.mk，翻到最下面找到全局宏定义，将想要数据入云的 sensor 的相关宏定义打开。这里以温度和湿度为例：

```
#GLOBAL_DEFINES += AOS_UDATA_SERVICE_BARO
#GLOBAL_DEFINES += AOS_UDATA_SERVICE_ALS
GLOBAL_DEFINES += AOS_UDATA_SERVICE_TEMP
GLOBAL_DEFINES += AOS_UDATA_SERVICE_HUMI
#GLOBAL_DEFINES += UDATA_SERVICE_PROXIMITY
#GLOBAL_DEFINES += AOS_UDATA_SERVICE_GPS
```

完成后在 AliOS Things Studio 中选择编译目标为 udataapp@developerkit (参考 2.3 章节), 然后打开根目录下的 .vscode/tasks.json, 找到 label 为 alios-studio: Make 配置项, 添加参数 dtc=1, 并进行手动保存, 如下:

```
{
  "label": "alios-studio: Make",
  "type": "shell",
  "command": "aos",
  "args": [
    "make",
    "uDataapp@developerkit",
    "dtc=1"
  ],
  "presentation": {
    "panel": "dedicated"
  }
},
```

在以上步骤都完成后, 进行编译烧写。设备启动后, 通过调试串口键入 netmgr 命令连上 wifi 网络:

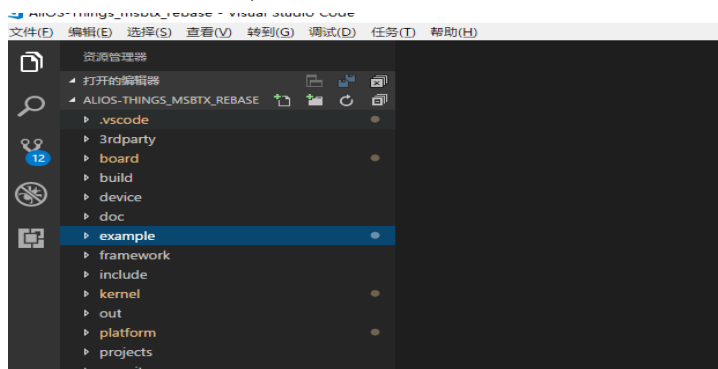
```
netmgr connect *ssid* *password* *open|wep|wpa|wpa2*
```


连接成功后, 串口会打印 "wifi_service_event", 并且 linkkit 例程会自动启动。打开云端的设备详情页面, 稍等片刻可以看到相关数据已经被记录在云端了:




代码主要路径: example\cameraapp\board\developerkit\camera_hal
代码编译:

- ### 1、 打开 Vissual studio Code， 打开代码目录



2、点击左下方的 , 输入 app 名字: cameraapp, 然后再输入 developerkit

3、点击  开始编译

4、编译完后, 点击  下载 bin 到开发板上, 最好断电重启一次。

cameraapp 涉及到按键、led、DCMI、SD、fatfs 等模块。

按键: 拍照用到的是 B 按键。采用上升沿触发中断方式。

```
hal_gpio_enable_irq(&brd_gpio_table[GPIO_KEY_3],  
                    IRQ_TRIGGER_RISING_EDGE, keyB_handle, NULL);  
keyB_handle 中断处理函数。
```

LED: 开发板 LED2, 低电平有效, 照片存储到 SD 成功后, led 会亮 2 秒。

DCMI: camera 的连接接口, 相关代码都是用 CubeMx 生成的, 数据的读取用到了 DMA2, 其中采用的 mode 是 DMA_CIRCULAR, 不要选择 DMA_NORMAL

SD、fatfs: SD 是以 fatfs 文件系统访问的。Camera 的照片是以 bmp 格式存储到 SD 的 DCMI 目录下。当 SD 没有插入时, 拍照时不会进入存储流程。相关函数是 camera_to_sd()。

3.3 其他例程陆续完善中

其他问题请参考 AliOS-Things WIKI:

<https://github.com/alibaba/AliOS-Things/wiki>