CoLink 单通道开关开发示例

版本 V0.9

关于本手册

本手册介绍了 colink 单通道开关开发过程。

章	标题	内容
第 1 章	开发简述	简要介绍 colink 开发目的和效果
第 2 章	开发步骤	详细介绍单通道开关的开发流程
第 3 章	配网示例	介绍配网部分开发流程
第 4 章	OTA 示例	介绍 OTA 部分开发流程
第 5 章	注意事项	介绍开发过程中的注意事项

发布说明

日期	标题	发布说明	编制	审核
2018.06.21	V 0.7	首次发布	王松	武鹏飞
2018.06.29	V 0.8	增加了编译烧录过程	王松	武鹏飞
2018.07.25	V 0.9	增加配网示例，增加 OTA 升级示例	王松	武鹏飞

目录

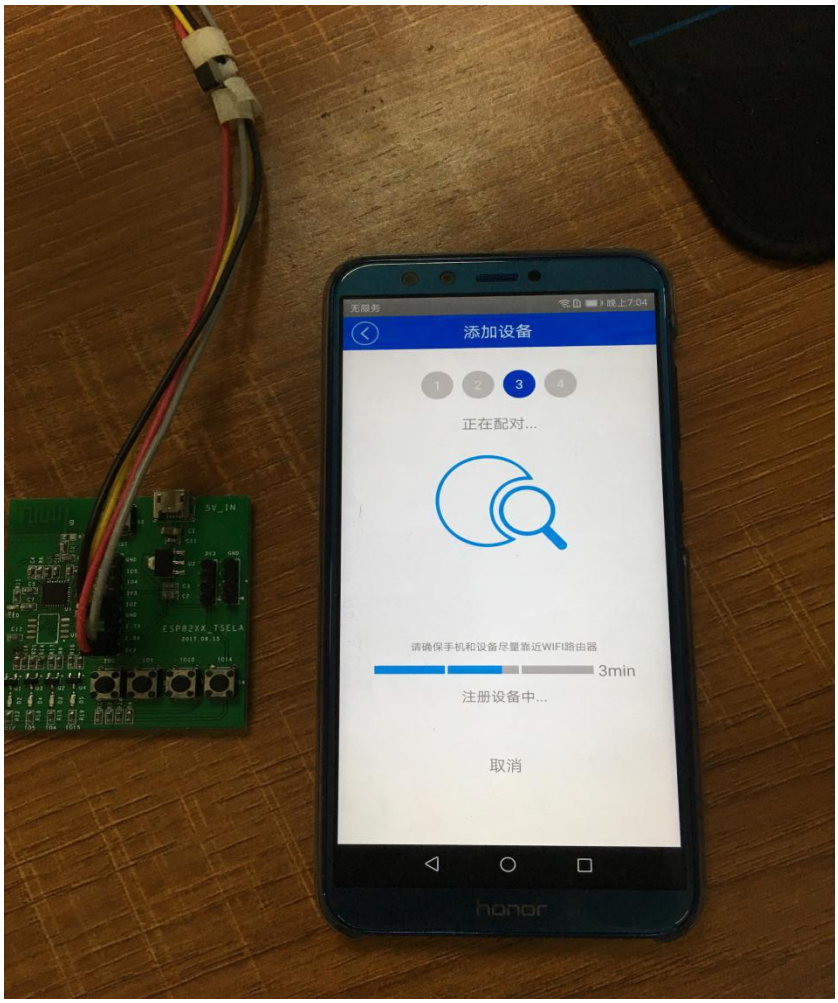
1. 开发简述

CoLink 是智能设备接入酷宅云平台的设备端应用开发框架，只需适配少量接口即可轻松接入酷宅云平台。开发者可以专注于产品功能的开发，缩短产品研发周期。本手册介绍了通过 colink 实现通过酷宅云平台和手机 APP 来控制一个单通道开关的目的。

1.1 功能说明

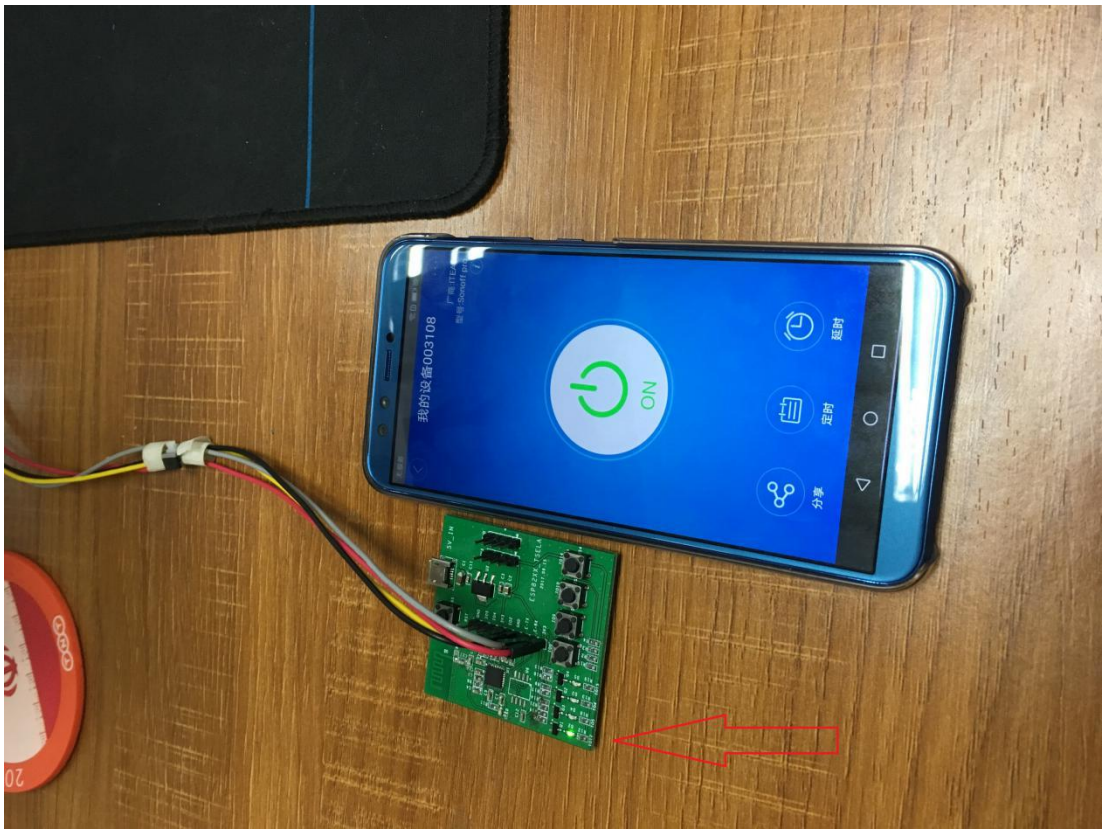
- 设备端可以通过按键短按来控制开关，并将信息传到云平台，在手机 APP 上同步小灯的开关状态
- 小灯可以通过手机APP的按钮来进行开关远程控制

以下为演示照片：

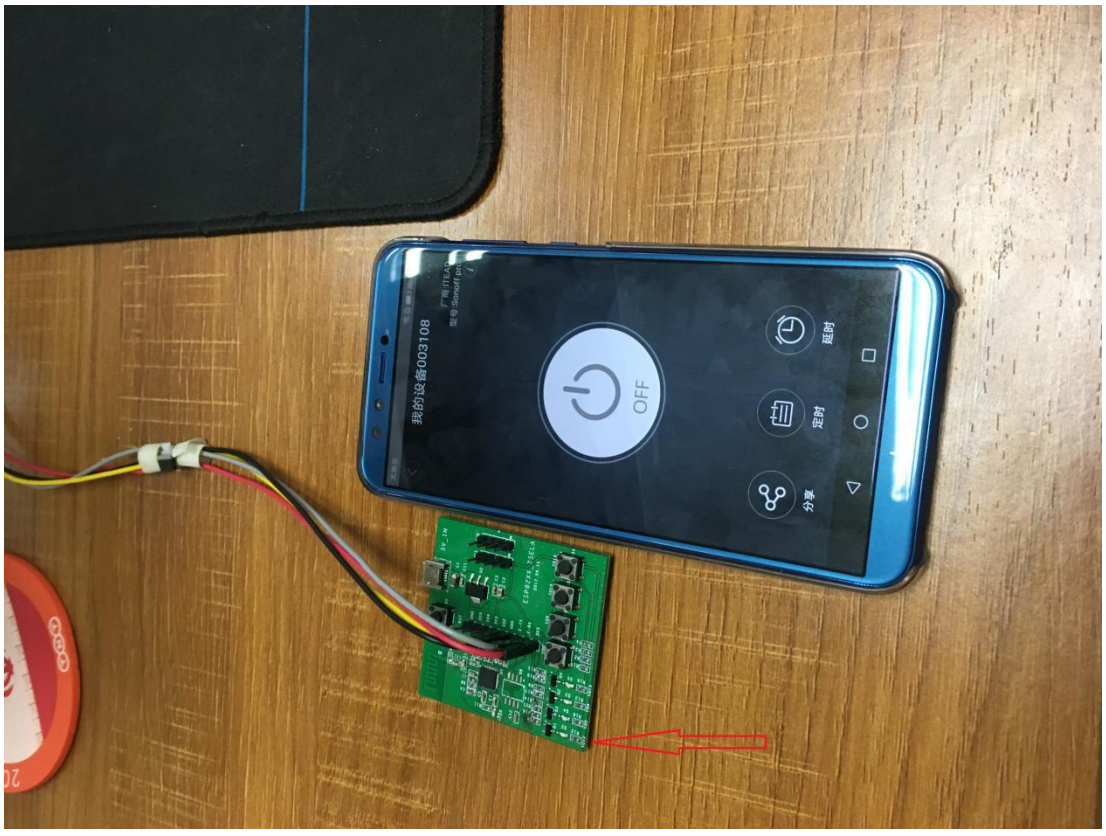


1、添加设备中

2、添加设备完成



3、设备和app同步开



4、设备和app同步关

1.2 使用说明

- 硬件平台：本次的产品演示是基于 ESP8266 的硬件平台的实现。
- 配网方式：ESP8266 提供的有 ESPTOUCH 和 SOFTAP 两种配网方式，此次采用的是 ESPTOUCH 的配网方式，用户需要根据自己的使用平台来实现配网部分。
- 接口适配：文件 colink_socket.h 和 colink_sysadapter.h 的接口需要用户根据自己使用的平台来进行部分的接口适配。
- 云平台相关：用户如果需要整个 demo 跑起来，需要向酷宅索要设备的数据，包括 deviceID、apikey；将正确的 deviceID、apikey 分别替换头文件 colink_define.h 的宏定义下#define DEVICEID "666666"和#define APIKEY "666666"的 "666666"。

2. 开发步骤

2.1 工具准备

- 交叉编译环境：ESP8266 芯片使用的编译环境是乐鑫官方提供的，这里不做重复说明，客户可以进入到乐鑫官网进行资料的下载，我们这里提供现在的下载地址：

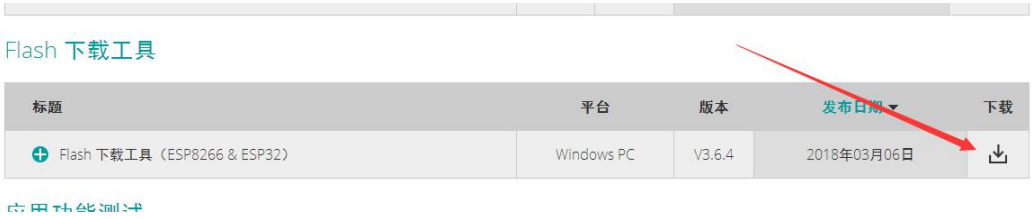
  https://www.espressif.com/sites/default/files/documentation/2a-esp8266-sdk_getting_started_guide_cn.pdf

  上面是 ESP8266 的入门指南下载地址，用户在搭建环境详细阅读它的 **3.3.1** 部分即可。

- 下载工具：基于乐鑫已经提供了多版本的下载工具，我们这里只针对目前最新的工具V3.6.4版本进行详细的介绍，其他版本的请用户自行参考使用方法，下面是乐鑫的工具集下载地址：

  https://www.espressif.com/zh-hans/support/download/other-tools

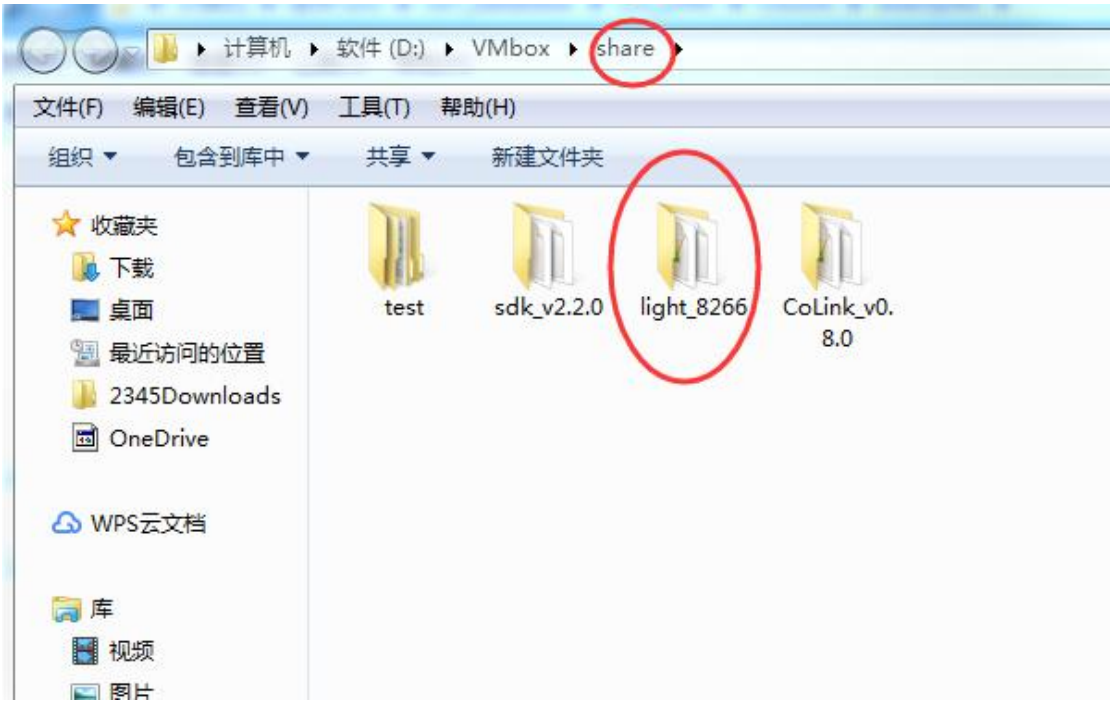
  进入到网站后下滑到Flash下载工具栏，点击下载即可。



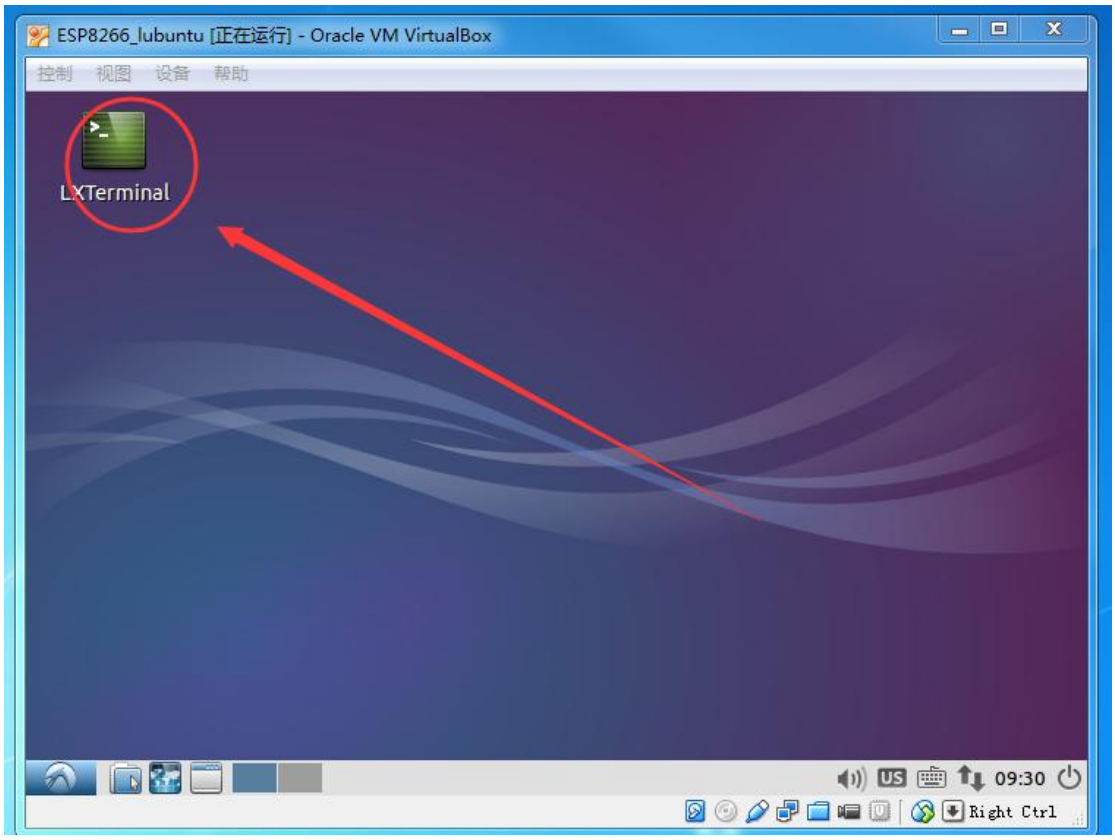
2.2 编译和烧录

2.2.1 编译过程

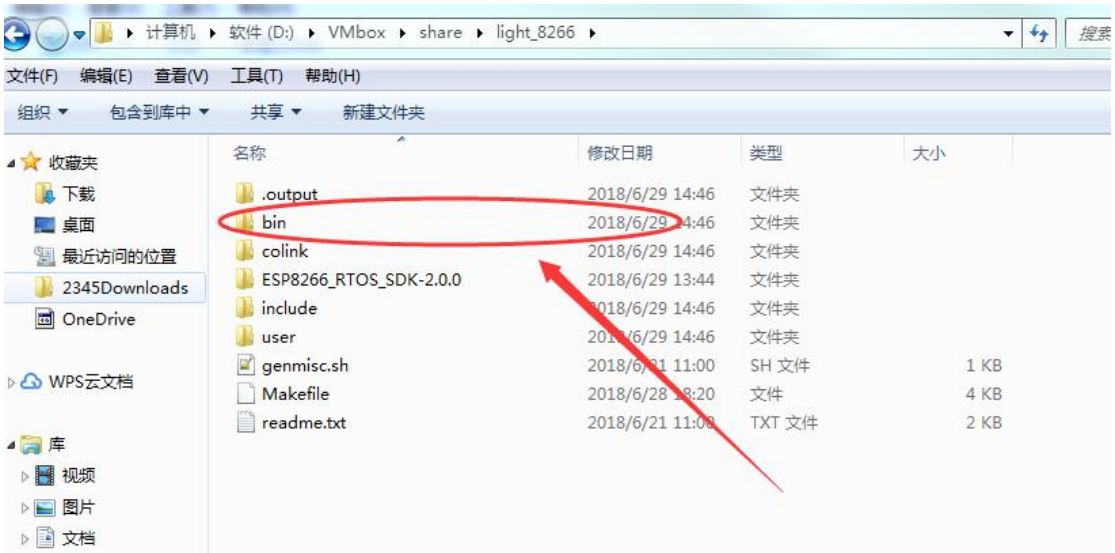
- 先将我们的 demo（light_8266）拷贝到 share 目录下（这个目录是我们在搭建环境的时候建立的，详情参考本文档 2.1 的内容）。



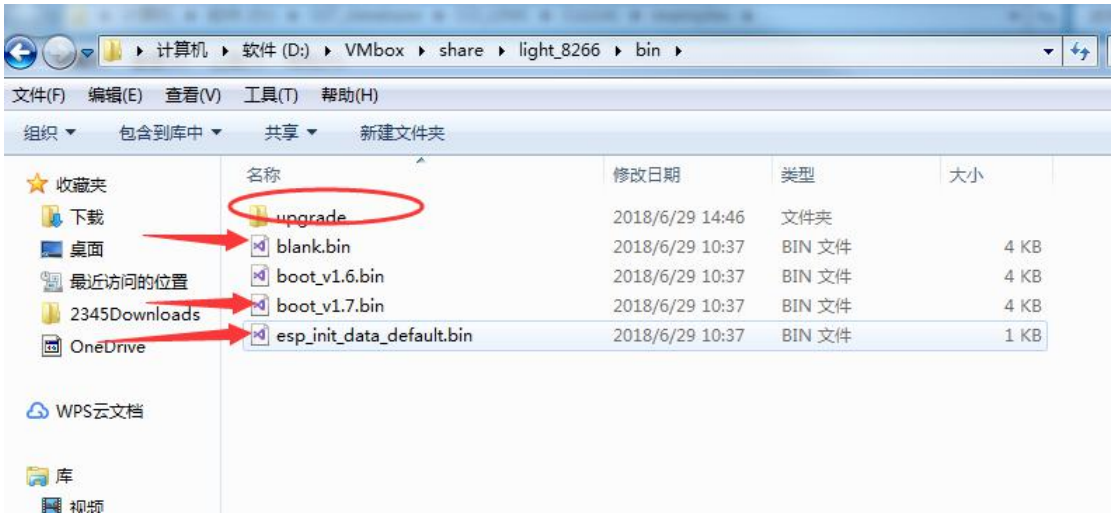
- 进入虚拟机，点开命令行图标 LXTerminal，输入 ./mount.sh 回车；输入密码 espressif 。

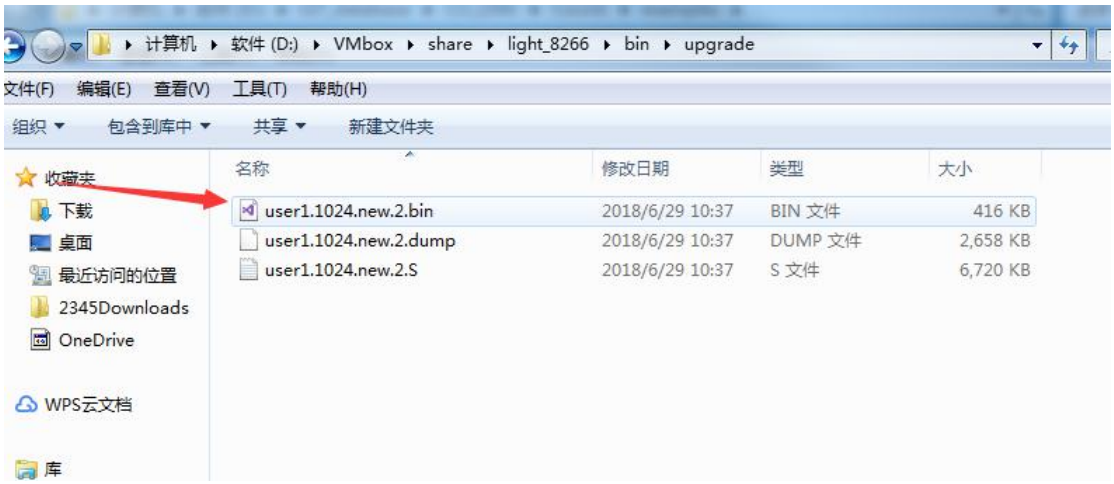


- 进入到 share 文件夹下的我们存放 demo 的目录进到里面，我们这里的是： cd /home/esp8266/Share/light_8266
- 输入 gen_misc.sh 编译即可,请根据自己的芯片选择编译。
- 编译生成的文件在目录 “ light_8266/bin ” 目录下。



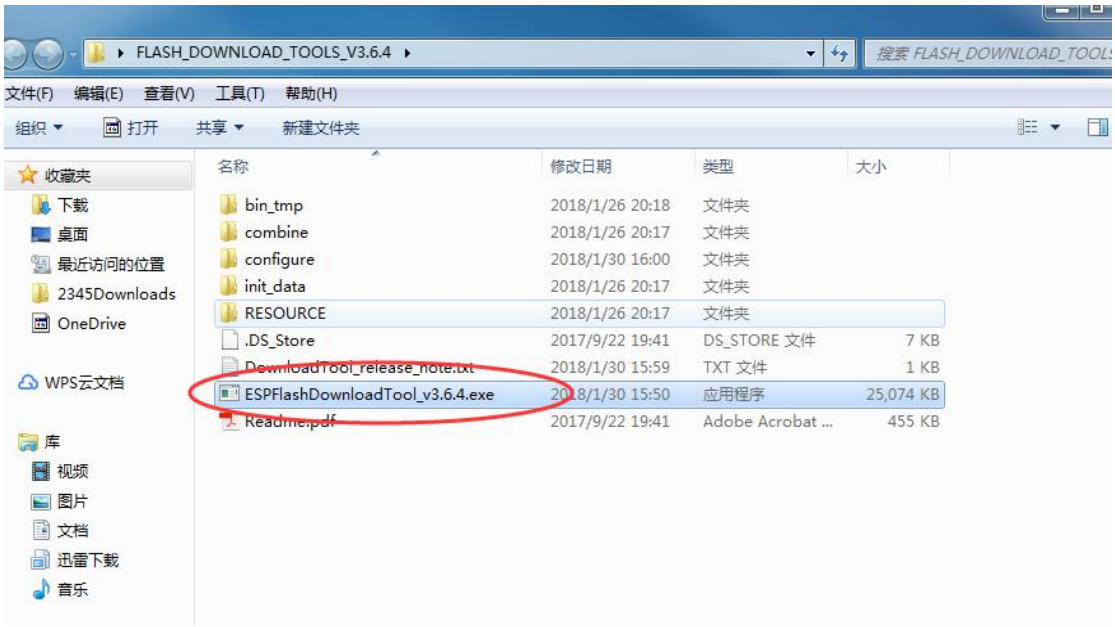
- 我们需要的是四个文件，分别是 boot_v1.7.bin、blank.bin、esp_init_data_default.bin、user1.1024.new.2.bin（这个文件在 upgrade 文件夹下）





2.2.2 烧录过程

- 烧录时，打开我们从乐鑫官网下载的烧录工具，本文档使用的是 V3.6.4 版本，详情参考本文档的 2.1 说明。
- 点击文件夹下的 .exe 文件。



- 根据自己使用的芯片选择对应的部分，这里我们使用的是 8285 的芯片，所以点击第二个（**用户根据自己实际芯片点击**）。

- 选择好我们需要的文件和地址，其中

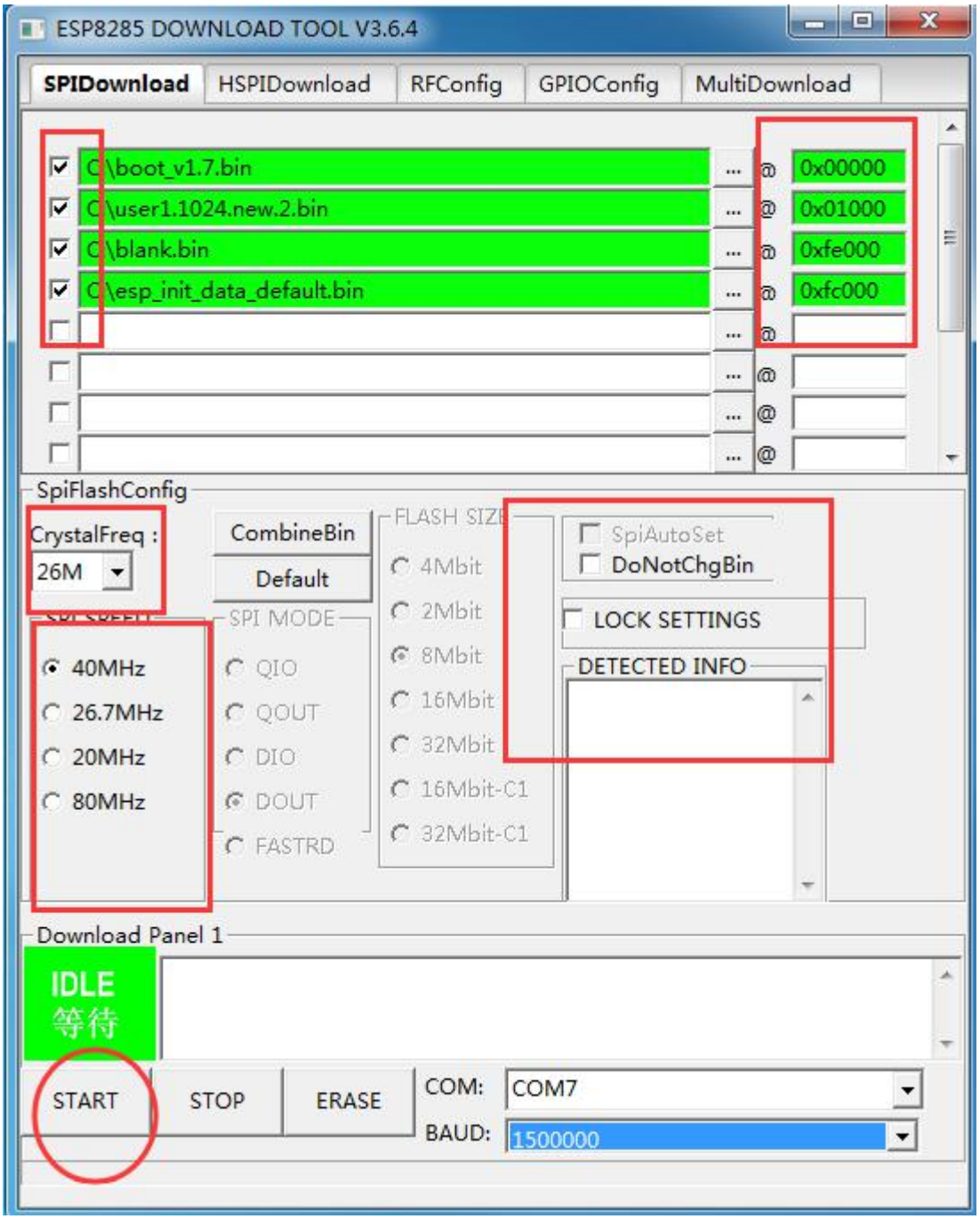
boot_v1.7.bin 对应地址 0x00000；

blank.bin 对应地址 0xFE000；

esp_init_data_default.bin 对应地址 0xFC000；

user1.1024.new.2.bin 对应地址 0x01000；

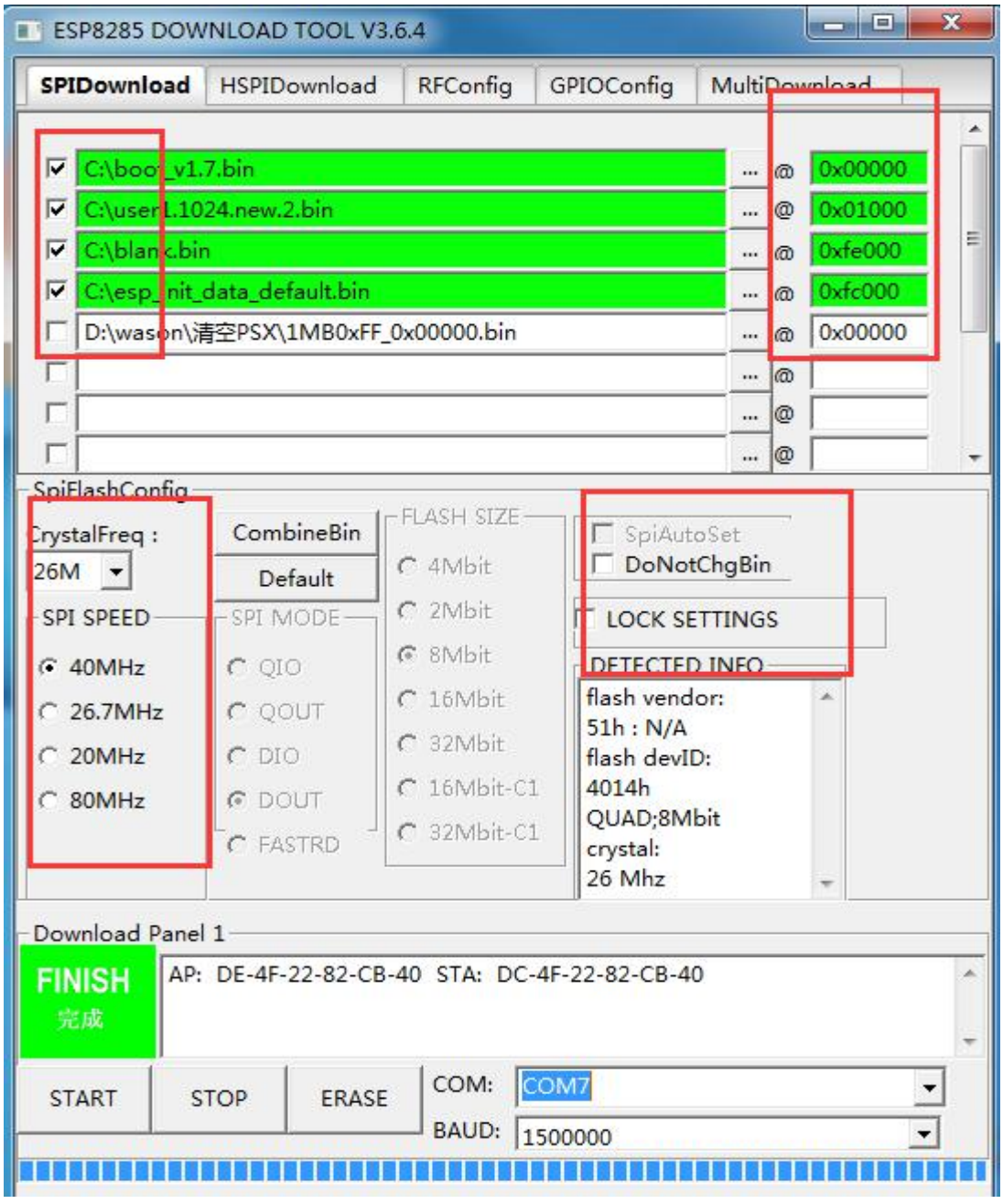
选择 CrystalFreq 为 26M，SPI SPEED 为 40MHz，其他不选，波特率的话尽量选择小，有的用户出现烧录不进去的原因是波特率调的太高，如果出现烧录失败，可以适当调整波特率试试。



- 接好自己所准备的测试板，选择正确的串口号，点击 START，按住测试板的 IO0 给测试板上电（或者按住 IO0 再按复位按键）即可烧录程序。
- 至此，我们就已经完成了基于 ESP8266/8285 芯片的整个 demo 的编译烧录流程，用户重启一下测试板就可以看到相关效果了。

2.2.3 烧录失败解决方法参考

- 烧录选项请严格按照我们的要求来，其中打红色方框的需要挺特别注意。



- 烧录过程可能会出现失败，这时候不要着急。1、检查线路，看线路是否松动；2、检查串口，看串口号是否正确，串口线是否接对；3、降低一下波特率，有的用户会出现波特率太高烧录失败；4、多试几次，可能自己的操作手法不对，我们的固

件经过多次测试，是可以被正确的烧录进入芯片和跑起来的。

2.3 设备基本运行流程

- 初始化串口，先清空发送缓冲区，定义好串口的波特率、数据位、奇偶校验、停止位，将串口 1 作为打印口。
- 初始化按键和小灯的 IO 口，将 GPIO12 设置为 GPIO 口，将 GPIO0 设置为 GPIO 口，并将 GPIO0 设置为输入端口。
- 建立任务定时器来检测按键触发情况。
- 任务定时器的回调函数，通过检测按键按下时长来判断长按和短按，并做出相应的动作，短按下会将开关状态改变，并将开关状态标志位改变，上报服务器设备的开关状态 ,使用 colinkSendUpdate 函数进行发送 ,输入参数为所需要发送的字符串，我们的协议是基于 json 格式的，所以通过 json 转化后进行发送。
- 初始化网络，根据我们使用的 ESP8266 平台，将设备设置为 station 模式，设置回调，直接进入到配网状态，得到 ssid 和 password 后连接 wifi，配网成功后，我们需要 定义两个变量

  ColinkDev *dev_data = NULL 和 ColinkEvent ev;并初始化

  ev.colinkRecvUpdate = colinkRecvUpdate;

  ev.colinkNotifyDevStatus = colinkNotifyDevStatus;

  colinkRecvUpdate 这个函数的作用是在接收到服务器发送过来的信息，他的参数是字符串，用户需要根据自己协议内容去解析所需要的信息，此次我们所用的协议是基于 json 格式的，所以采用 json 来解析，并根据内容来动作开关；

  colinkNotifyDevStatus 这个函数是设备在线和离线状态发生改变时的回调函数，

输入参数为枚举类型 ColinkDevStatus。

- 配置 dev_data 变量的内容，deviceid 和 apikey 需要向酷宅索要（前面 1.2 使用说明已经说过）；并配置好 APP 下发给设备的域名，定义好固件版本号，由于此次我们使用了 mbedtls，客户可以根据需要选择使能失能，使能时将#define SSLUSERENABLE 和#define COLINK_SSL 和#define COLINK_VERIFY 这三个宏开放出来，不需要关闭即可，在 colink0.9.0 中我们增加了设备类型，请根据自己实际设备类型选取 dev_data->dev_type，此时我们就可以调用 colinkInit 这个函数进行初始化了，这个函数的输入参数即为 dev_data 和 ev；最后一步我们就要周期性调用 colinkProcess 这个函数就可以了，这个函数的调用周期我们的建议是 25-50ms。至此，我们就实现了整个 colink 架构下的单通道开关的实现了。

3. 配网示例

3.1 AP 模式配网

3.1.1 配网说明

- 我们设置的 AP 配网是在 ESPTOUCH 状态下长按 5s 的小灯按键进入的 ,所以在使用 AP 配网时需要是设备处于 ESPTOUCH 模式

3.1.2 配网流程

- 在按键长按 5s 后建立一个进入 AP 配网的函数 enterSettingSelfAPMode() ,建立好 AP 模式的任务 colinkSelfAPTask
- 配置号 AP 配网结构体的各个参数 struct softap_config
- 关闭 smartconfig，断开当前的 wifi 连接，将设备操作模式修改为 STATIONAP_MODE，并设置号 sap 的 IP 信息，此时即可进入到和 app 的交互部分
- 在和 APP 交互的任务 colinkSettingTask 里 ,我们采用 socket 编程建立网络连接，这里我们封装有库函数来和 app 进行数据交互，用户需要做的事情是将 app 发过来的数据传入，再将得到的信息取出来
- 第一步是建立好 socket 连接，第二步是将 app 发过来的数据传入到 colinkLinkParse 这个函数中，这个函数的返回值需要先是 COLINK_LINK_RES_DEV_INFO_OK，我们将 colinkLinkParse 中输出的数据通过 socket 发送，再次得到 COLINK_LINK_GET_INFO_OK，再次将 colinkLinkParse

中输出的数据通过 socket 发送，此时就可以调用 colinkLinkGetInfo 来拿到我们所需要的信息了

- app 交互结束后，我们需要将设备设置为 station 模式，去连接所拿到信息的 wifi

3.2 ESPTOUCH 模式配网

3.2.1 配网说明

- 我们设置的 ESPTOUCH 配网是在上电就进行的
- 在上电初始化后就可以直接使用我们 ESPTOUCH 模式配网

3.2.2 配网流程

- 首先设置我们设备工作在 station 模式，设置号 smartconfig 的类型 smartconfig_set_type(SC_TYPE_ESPTOUCH_AIRKISS);并开启 smartconfig，在它的回调中拿到 ssid 和 password 连接 wifi ，smartconfig 结束后我们会去和 app 交互，此部分和 AP 配网一样，用户只需要建立起 tcp 连接后就可以调用我们的接口处理，不需要管具体的数据交互，带来开发上的方便

4. OTA 示例

4.1 OTA 升级说明

- OTA 升级是设备在线向升级服务器请求下载新的固件，并校验存储在 flash 中并重新启动运行新固件的方式
- 用户首先需要把 user.bin 和 user2.bin、设备的 deviceID、固件升级的版本号提供给酷宅工程师，由工程师在服务器上将升级包配置好后才可以进行固件升级

4.2 OTA 升流程

- 首先当我们在 app 端点击升级之后，服务器会向我们的设备发送一段请求，这是会产生一个回调 colinkUpgradeRequest ，这个函数的参数 ColinkOtaInfo 内的内容是我们新固件的信息，通过查询我们此时运行的固件，得到需要下载的固件的链接，端口号，IP 地址和自身去储存部分
- 根据得到信息建立好 tcp 链接，并到服务器下载固件，将我们下载下来的固件进行校验并储存到 flash 相应的扇区，完成之后调用 colinkUpgradeRes 将设备升级完成的信息返回给服务器即可，中途出错的情况可以传输相应的参数给服务器来报错
- 编译的时候分别运行 light_8266 文件中的 gen_misc.sh,根据自己的芯片选择好配置，得到两个 bin 文件，user1.1024.new.2.bin、user2.1024.new.2.bin，这两个 bin 文件就是我们的升级固件

5. 注意事项

5.1 开发前的注意事项

- 与使用平台相关的函数实现需要用户自己配置，具体到 colink_socket.h 和 colink_sysadapter.h 两个头文件。
- 需要提前向酷宅索取 deviceID 和 apikey，替换掉 colink_define.h 头文件中 DEVICEID 和 APIKEY 的宏定义。
- 配网方式需要用户根据自己平台提供的方式去实现，获取到的服务器的域名需要用户自己进行有效存储以提供给 colinkInit 函数的参数使用。
- colinkInit 的参数有两个，一个为 dev_data，包含设备信息和服务器域名，另一个为 ev，它的函数实现由用户根据自己需求编写，ev.colinkRecvUpdate 为接收到服务器的信息回调，输入的参数即为服务器发送过来字符串的信息，ev.colinkNotifyDevStatus 为设备在线状态改变时的回调函数，根据用户自己的需求自己编写。
- 在初始化调用 colinkInit 进行初始化后，我们需要周期行调用 colinkProcess 函数即可，这个函数的调用周期为 25-50ms。
- 设备状态改变后需要向服务器发送信息要调用 colinkSendUpdate 函数，这个函数的输入参数为字符串，即为用户需要向服务器发送的字符串信息。

免责申明和版权公告

本文中的信息，包括供参考的 URL 地址，如有变更，恕不另行通知。文档“按现状”提供，不负任何担保责任，包括对适销性、适用于特定用途或非侵权性的任何担保，和任何提案、规格或样品在他处提到的任何担保。本文档不负任何责任，包括使用本文档内信息产生的侵犯任何专利权行为的责任。本文档在此未以禁止反言或其他方式授予任何知识产权使用许可，不管是明示许可还是暗示许可。Wi-Fi 联盟成员标志归 Wi-Fi 联盟所有。ESP 商标为乐鑫公司注册商标文中提到的所有商标名称、商标和注册商标属其各自自所有者的财产，特此声明。