

**QuePython**

**云喇叭软件设计文档**

**LTE系列**

版本：QuecPython云喇叭软件设计文档\_V1.0

日期：2022-11-24

 状态：临时文件

www.quectel.com

|  |
| --- |
|  |
|  |

上海移远通信技术股份有限公司始终以为客户提供最及时、最全面的服务为宗旨。如需任何帮助，请随时联系我司上海总部，联系方式如下：

上海移远通信技术股份有限公司

上海市闵行区田林路1016号科技绿洲3期（B区）5号楼 邮编：200233

电话：+86 21 51086236 邮箱：[info@quectel.com](mailto:info@quectel.com)

或联系我司当地办事处，详情请登录：

<http://quectel.com/cn/support/sales.htm>

如需技术支持或反馈我司技术文档中的问题，可随时登陆如下网址：

<http://quectel.com/cn/support/technical.htm>

或发送邮件至：[support@quectel.com](mailto:support@quectel.com)

**前言**

上海移远通信技术股份有限公司提供该文档内容用以支持其客户的产品设计。客户须按照文档中提供的规范、参数来设计其产品。由于客户操作不当而造成的人身伤害或财产损失，本公司不承担任何责任。在未声明前，上海移远通信技术股份有限公司有权对该文档进行更新。

**版权申明**

本文档版权属于上海移远通信技术股份有限公司，任何人未经我司允许而复制转载该文档将承担法律责任。

版权所有 ©上海移远通信技术股份有限公司 2019，保留一切权利。

***Copyright ©*** ***Quectel Wireless Solutions Co., Ltd. 2019.***

# 文档历史

**修订记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **作者** | **变更表述** |
| 1.0 | 2022-11-24 | Pawn | 初始版本 |

# 目录

[文档历史 3](#_Toc121572835)

[目录 4](#_Toc121572836)

[1 前言 5](#_Toc121572837)

[2 系统框架 6](#_Toc121572838)

[2.1. 硬件系统框架 6](#_Toc121572841)

[2.2. 软件系统框架 6](#_Toc121572842)

[3 关键组件 8](#_Toc121572843)

[3.1. EventMesh 9](#_Toc121572844)

[3.2. AudioManager 11](#_Toc121572845)

[3.3. ConfigStoreManager 12](#_Toc121572846)

[3.4. LteNetManager 13](#_Toc121572847)

[3.5. DeviceInfoManager 15](#_Toc121572848)

[3.6. OtaManager 15](#_Toc121572849)

[3.7. AliYunManage 16](#_Toc121572850)

[3.8. GLight 17](#_Toc121572851)

[3.9. DeviceActionManager 17](#_Toc121572852)

[4 系统初始化流程 19](#_Toc121572853)

[5 业务流程 20](#_Toc121572854)

[6 功能示例 21](#_Toc121572855)

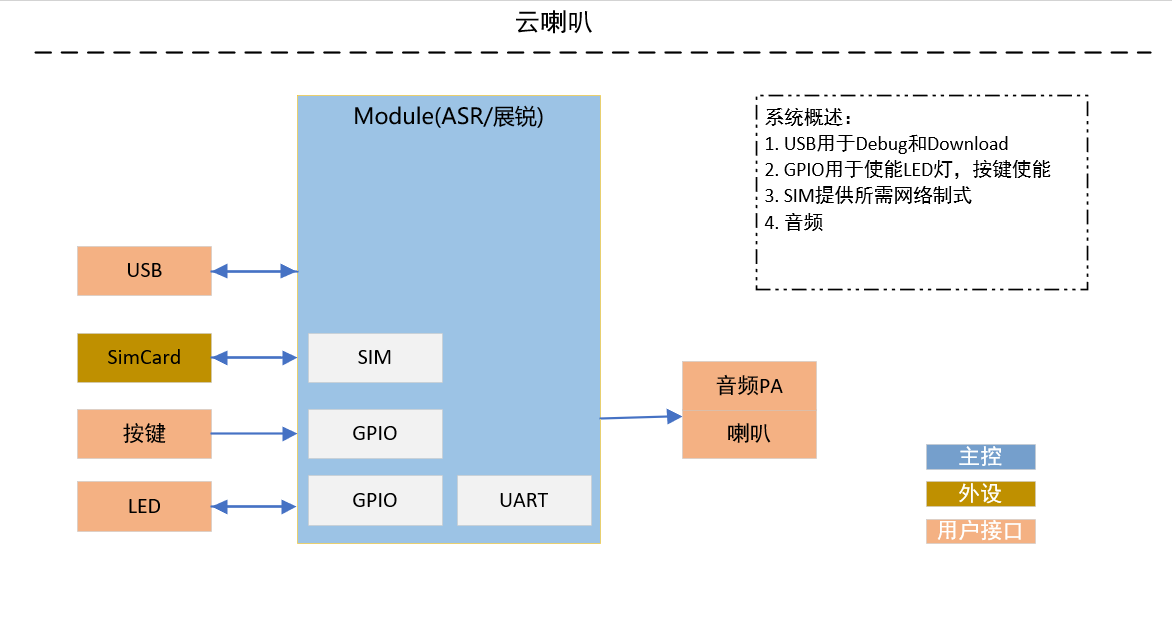
# 

1. 前言

此文档用于描述云喇叭软件设计框架,包含核心组件功能描述，系统初始化流程的介绍等，方便更快理解本框架。

1. 系统框架
3. 1. 硬件系统框架

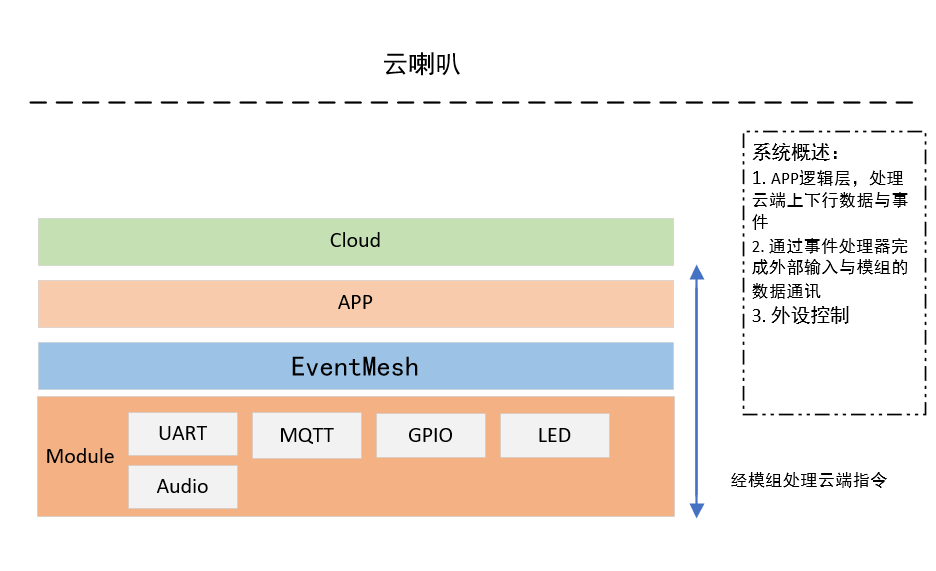
系统框架如下，Module侧支持SIM，GPIO，UART等功能。



* 1. 软件系统框架

软件系统框架如下:

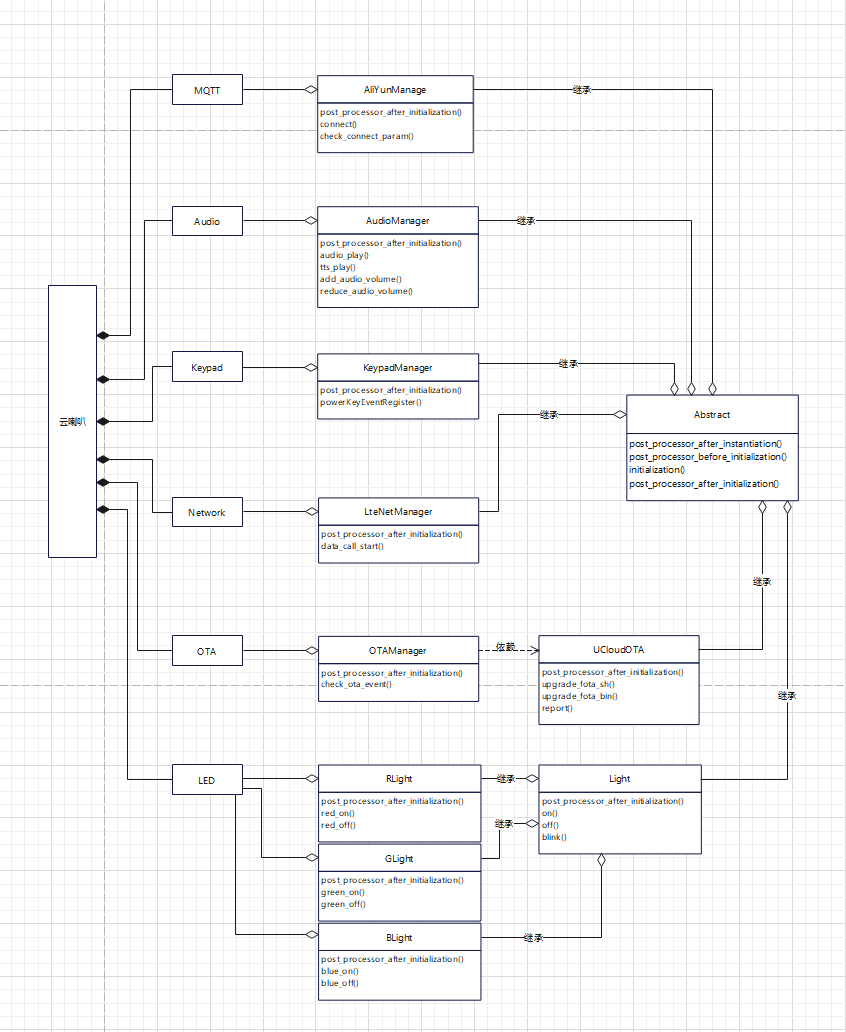
1. APP层实现处理电表核心业务，解析上下行数据
2. EventMesh为事件处理器，通过支持事件订阅发布的机制来完成功能流转
3. Module侧接收外部事件或数据通过EventMesh驱动来处理执行



1. 关键组件

组件对象关系描述:

下图用于描绘项目软件代码中各组件对象之间的依赖关系与继承关系，图中以云喇叭作为总对象，将云喇叭所依赖的功能对象联系到一起，核心的对象拥有哪些方法，依赖哪些功能类，具体如下图所示：



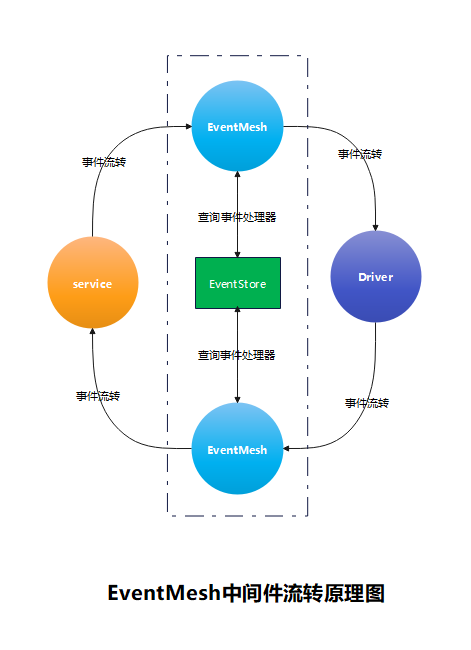
* 1. EventMesh

1. 功能描述：

EventMesh是一种动态基础中间件, 在事件驱动架构语境中,事件指的是系统中的变更、操作或观察,他们会生成通知, 然后响应到各个对事件做出响应的处理器函数中

1. 实现原理：

详细说明请参照EventMesh中间件设计文档，以下而为简介。



注解: 服务和驱动通过EventMesh进行数据通信, 所有的Event都是从EventStore里面过来的, 这里注意EventStore只是一种概念, 而不是真正意义上对外展示的东西

1. 订阅事件

文本

描述已自动生成

1. 发布事件



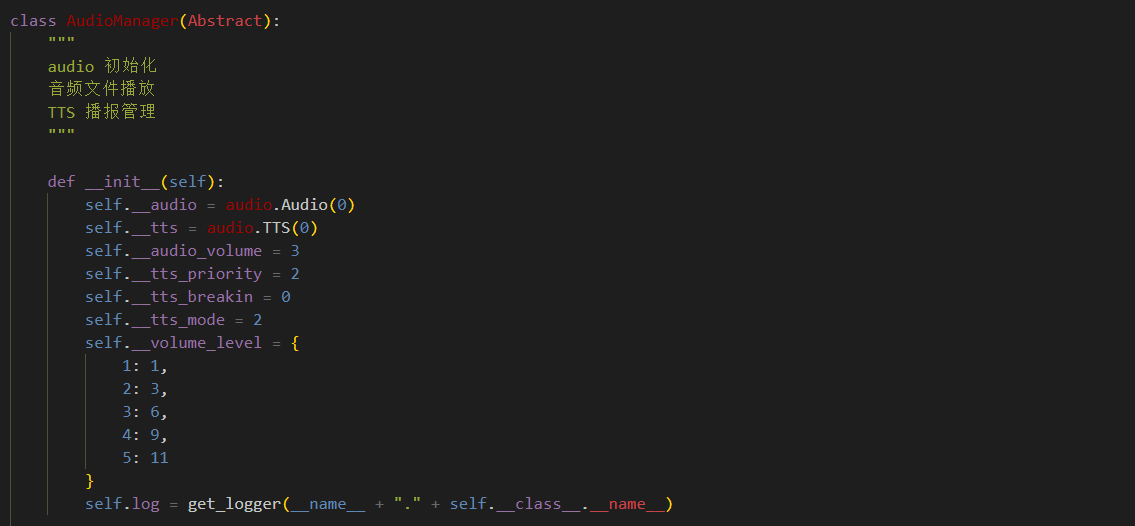
* 1. AudioManager

1. 功能描述：

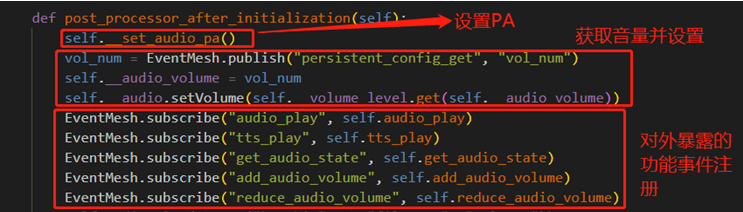
控制设备音频输出，包含TTS语音播报，音频文件播放，音频音量设置等功能。

2. 实现原理：

1. 创建类方法会在\_\_init\_\_方法内完成audio功能的初始化，将原始音量等级简化成5个等级，方便使用。



2. 初始化类方法后将对外暴露的方法通过事件注册到EventMesh中，完成音量初始化设置。



3. Audio播放音频文件，通过QuecPython提供的audio API直接使用，API详细使用描述参考wiki文档。

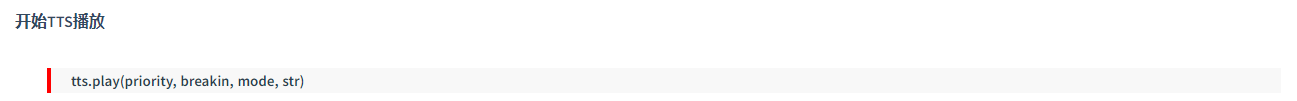
图形用户界面, 应用程序

描述已自动生成

文本

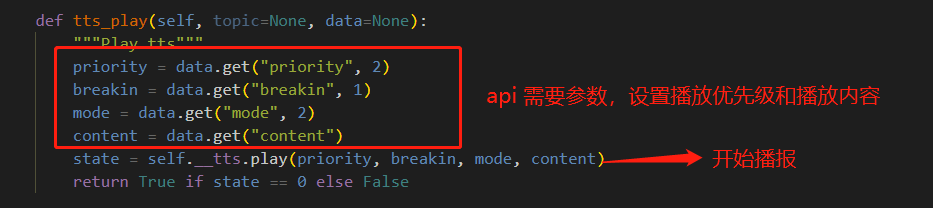
描述已自动生成

4. TTS播放输入内容，通过QuecPython提供的TTS API直接使用，API详细使用描述参考wiki文档。



图形用户界面, 应用程序

描述已自动生成

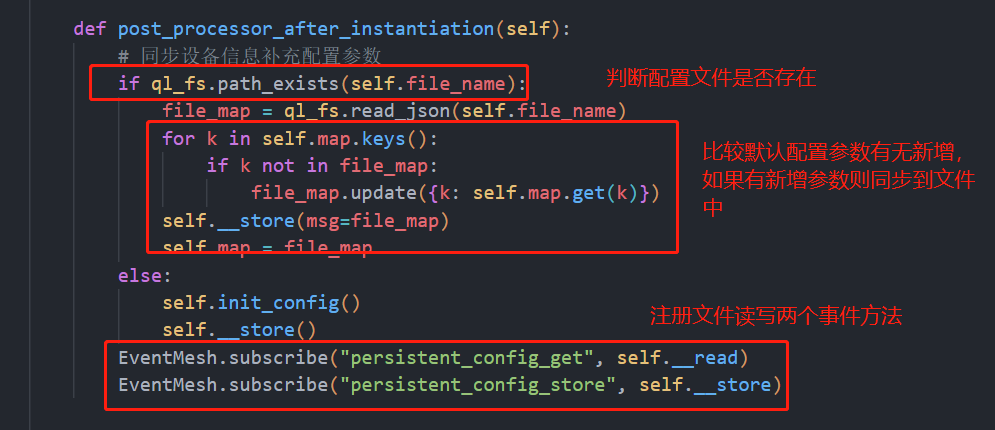


* 1. ConfigStoreManager

1. 功能描述：

该功能主要用于处理设备参数持久化保存，提供参数读取于参数更新的方法，通过该方法与模组文件系统交互。

1. 实现原理：
2. 初始化类方法时会判断配置文件是否存在，不存在则创建一个JSON文件，文件创建成功后将默认参数写入文件中。
3. 若文件已存在，则会比对默认参数有无新增，若有新增同步更新到配置文件
4. 该类方法对外通过EventMesh注册读取和写入两个事件



* 1. LteNetManager

1. 功能描述：

该功能主要用于模组网络初始化以及网络状态管理，设备默认为自动拨号注网，如需调整为手动拨号或者需要使用设置的APN进行拨号请参考wiki文档进行修改，下面做简要描述。

2. 实现原理：

1. 该类方法初始化时会等待设备注网完成，且注册网络变化回调函数，通过checkNet API返回值判断设备找网状态，找网成功后会通过事件发布的方式启动TCP连接，若找网失败则会尝试重新找网。

2. checkNet API可在wiki中心查找到详细描述：<https://python.quectel.com/wiki>

文本

描述已自动生成

3. 切换到手动拨号

图形用户界面, 应用程序

描述已自动生成

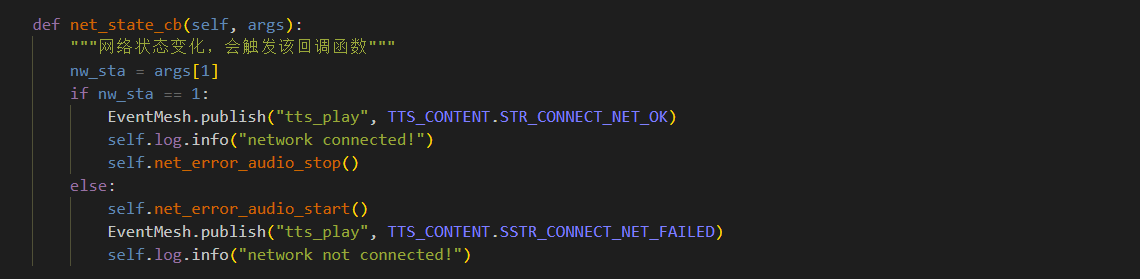
4. 配置用户APN



5. 网络状态异常处理

当设备网络状态发生变化时我们可以通过注册回调的方式来通知到应用层，如下所示：





网络状态出现异常重连的示例：

网络异常后会先尝试使用Cfun切换来重新找网，若cfun失败可考虑重启模组，若需要做次数限制

则通过文件记录一个值来控制。

文本

描述已自动生成

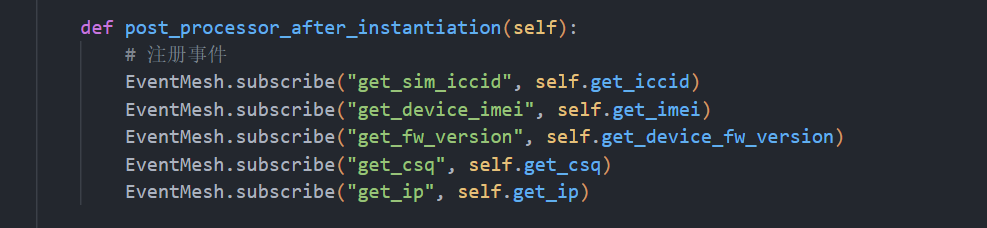
* 1. DeviceInfoManager

1. 功能描述：

该功能用于获取设备的一些基础信息，例如设备IMEI，SIM card的ICCID等。

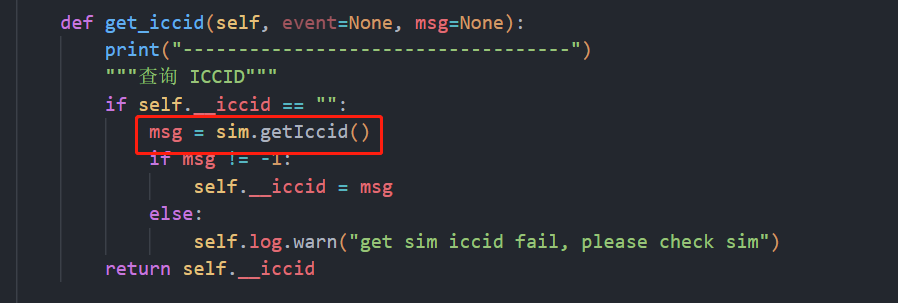
2. 实现原理：

1. 类函数初始化时会将获取设备信息的函数注册成事件对外提供，可直接通过事件发布的方式使用。



2. 类似查询iccid

使用sim API 获取当前sim card的iccid



3. 查询设备信息的API 使用详细描述可参考wiki文档：https://python.quectel.com/wiki

* 1. OtaManager

1. 功能描述：

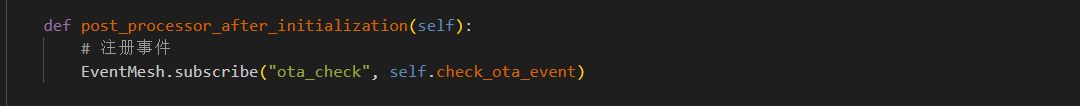
该功能用于对APP应用程序代码通过远程升级更新

2. 实现原理：

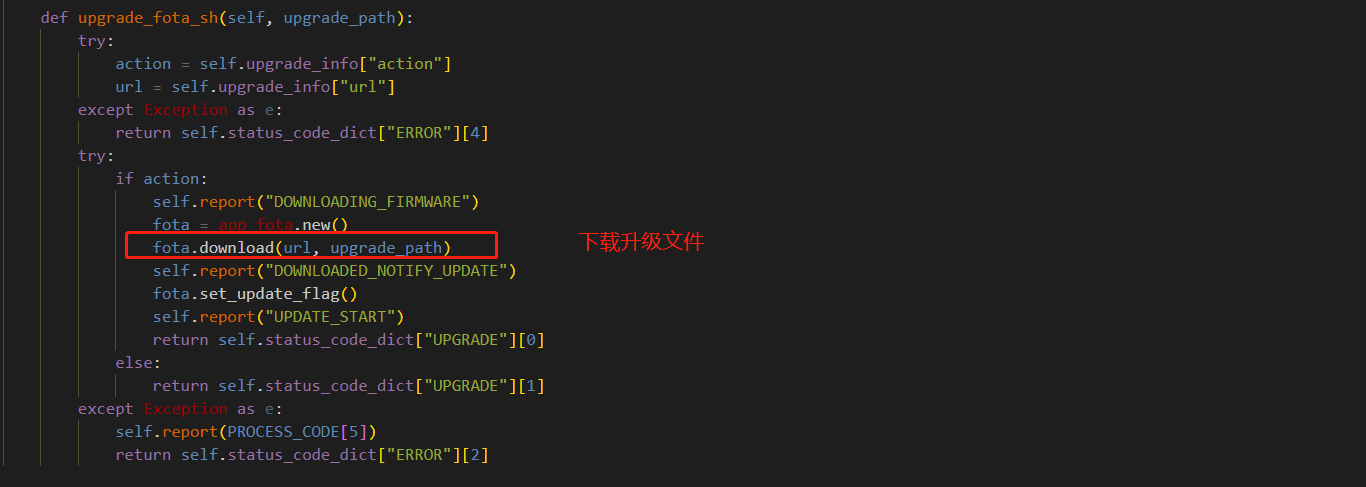
1. 使用移远OTA平台进行升级。

2. 在平台创建升级任务后设备上电注网后通过API检测是否有升级任务

3. 对外提供OTA升级检测接口，通过事件发布的方式调用该方法。



3. 存在升级任务后调取下载接口，下载完成后设备会重启完成升级



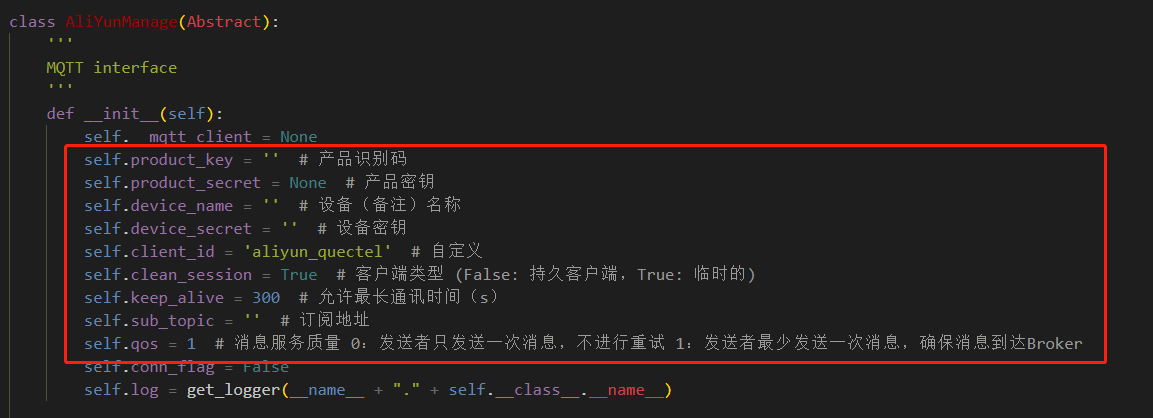
* 1. AliYunManage

1. 功能描述：

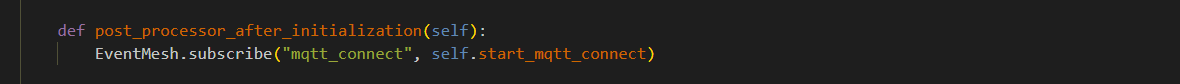
设备通过MQTT协议连接到阿里云IOT平台，提供连接，断开连接，数据上下行等功能。

2. 实现原理：

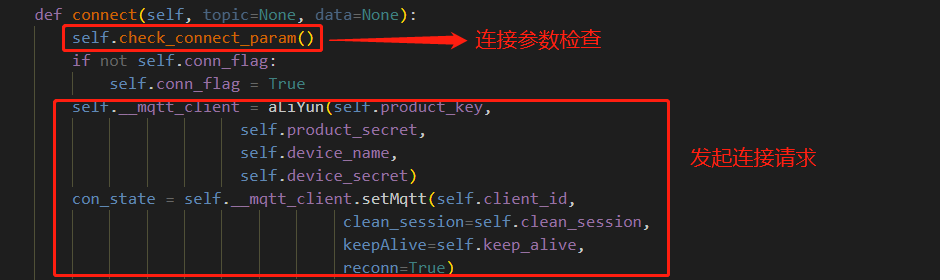
1. 创建类方法会在\_\_init\_\_内完成类属性的初始化，类属性包含连接云平台的三元组，保活时间等。



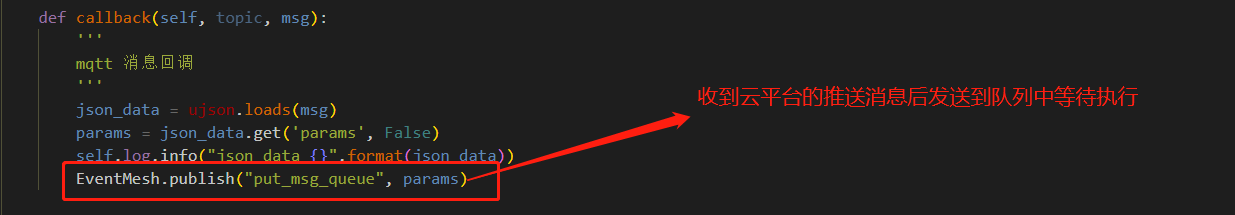
2. 初始化类方法后将对外暴露的方法通过事件注册到EventMesh中。



3. 发起连接请求。



4. 下行数据回调函数。



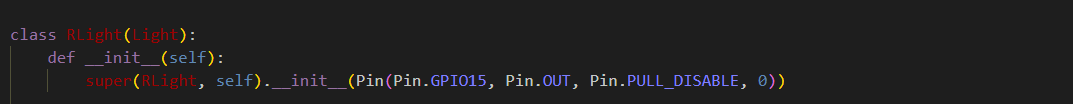
* 1. GLight

1. 功能描述：

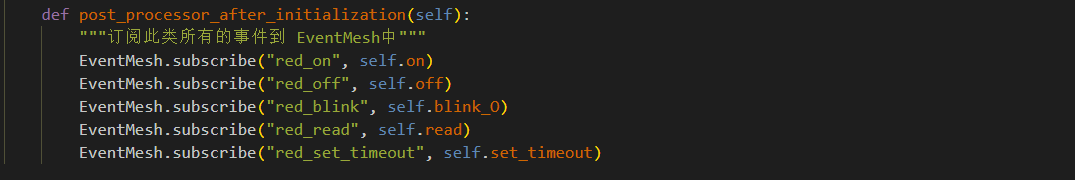
设备Led灯状态管理，通过GPIO使能控制Led的开关和闪烁动作。

2. 实现原理：

1.创建类方法会在\_\_init\_\_内完成类属性的初始化和led灯GPIO管脚的初始化。



2. 初始化类方法后将对外暴露的方法通过事件注册到EventMesh中。



3. 使能GPIO管脚控制灯。

形状

中度可信度描述已自动生成

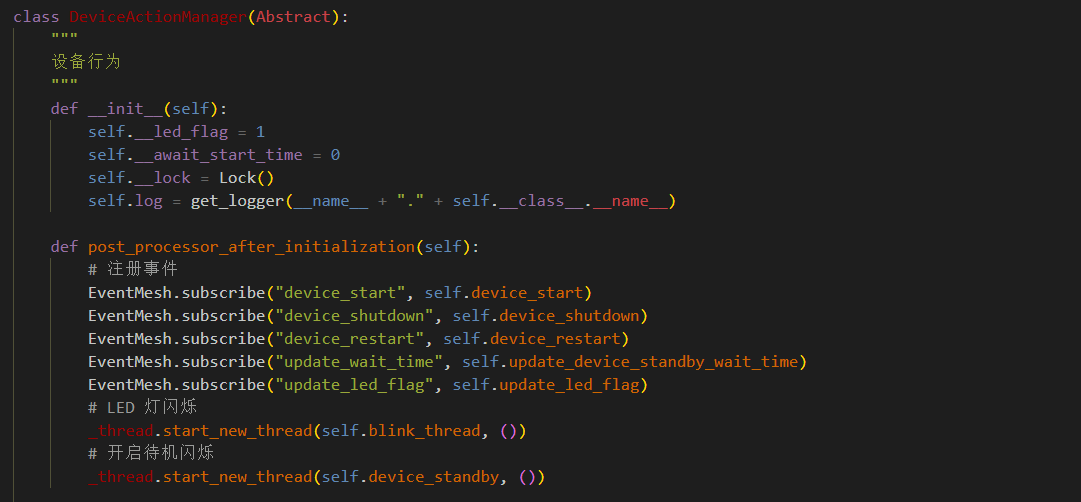
* 1. DeviceActionManager

1. 功能描述：

设备状态控制，设备开关机，重启，待机状态，外设控制。

2. 实现原理：

1. 初始化类方法后将对外暴露的方法通过事件注册到EventMesh中。

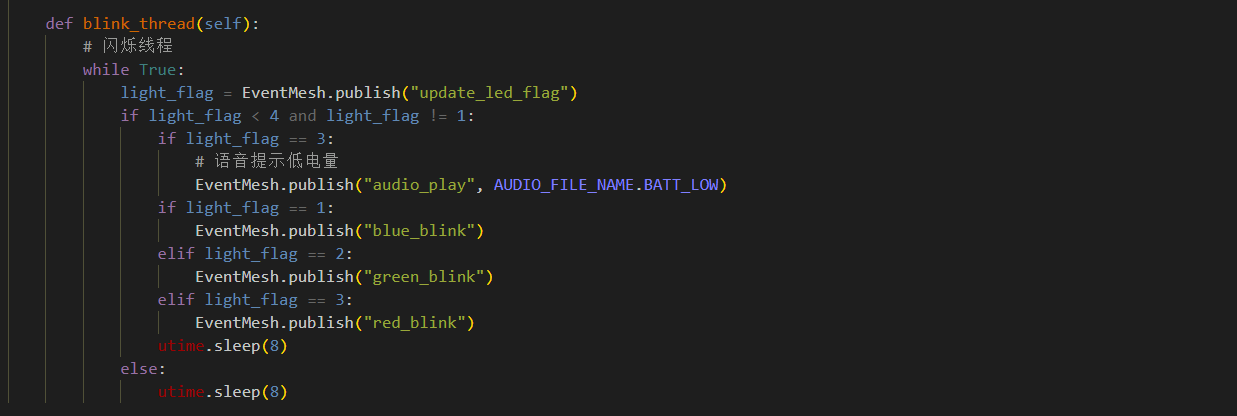


3. 设备待机状态。

图片包含 文本

描述已自动生成

4. 设备LED灯控制任务

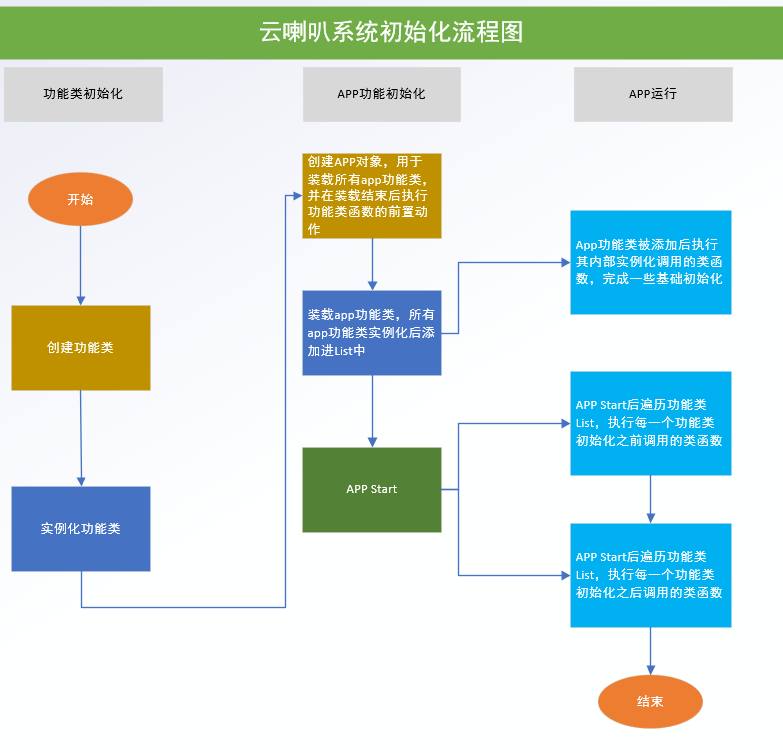


5. 设备开关机动作。

文本

描述已自动生成

1. 系统初始化流程



系统初始化流程说明：

1. 云喇叭所有功能类进行初始化。
2. 所有的类方法都会约定必须有初始化前后或实例化前后要完成的事件注册或功能处理，所以我们通过一个APP类将所有的类方法在装载时和start时会将每个类方法的初始化前后动作执行完。
3. 业务流程

图示

描述已自动生成

1. 功能示例

本章节介绍如何基于项目代码进行二次开发，例如新增功能，修改代码功能逻辑等，可以参照下面的示例帮助快速上手，熟悉代码流程。

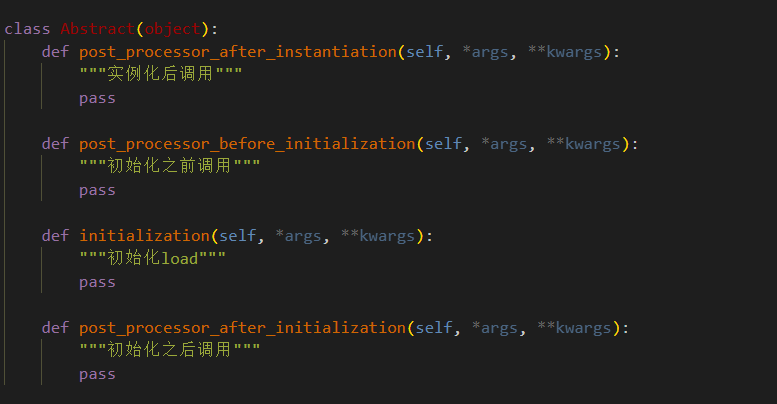
在准备调整代码前需要先了解当前代码中功能是如何流转运行的，下面会做步骤描述：

1. 首先整个项目中实现的功能函数都采取的是使用EventMesg来进行事件驱动，通俗的讲就是我们现在实现了一个功能功函数，然后将这个函数定义一个事件名称，例如查询设备IMEI，我们将事件名称与对应的事件函数以键值对的方式做映射(”Get\_Imei”: Function),这种创建映射关系的动作理解成订阅事件。而EventMesh会保存这些订阅事件，当我们需要查询设备IMEI的时候只需要去发布Get\_Imei这个事件即可拿到返回结果。这样我们每个功能都可以独立开发，独立调试，达到解耦合的效果。
2. 新增一个功能类函数，我们先实现该功能函数，如图：



如上图中我们首先创建了一个功能类方法，这个类方法继承了一个公共基类Abstract，这个基类是用来约束开发者在功能实现上保持一致性；然后我们在类方法中重新实现了基类的post\_processor\_after\_instantiation()方法，为什么要实现该方法呢，因为我们新增的这个功能在实现完之后需要加入到APP中去注册启动，APP类会将每一个继承于Abstract类的功能类下初始化方法调用执行，也就是post\_processor\_after\_instantiation()方法，所以我们需要在该方法中将需要对外暴露的方法订阅到EventMesh中去。

Abstract基类：



APP类：

日程表

描述已自动生成

3.将新增的类方法加入到main.py入口函数中：

1.导入新增的类方法：

这里的usr.msg指的是usr（用户）分区下面的mgr.py文件，因为我的类方法是是现在mgr.py中的

文本

描述已自动生成

2.将类方法添加到App类中：

文本

描述已自动生成

1. 使用新增的类方法：

1.前几步中我们已经完成的类方法的新建，功能实现以及事件注册，添加到App中启动，那么接下来介绍怎么使用新增的类方法：

文本

描述已自动生成

通过事件发布的方式找到已经订阅好的事件，执行对应的事件函数。