QuecPython
软件注意事项及 FAQ

LTE Standard 模块系列

版本：1.0.0

日期：2021-02-02

状态：临时文件

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

或联系我司当地办事处，详情请登录：http://www.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://www.quectel.com/cn/support/technical.htm 或发送邮件至：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。因未能遵守有关操作或设计规范而造成的损害，上海移远通信技术股份有限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。

免责声明

上海移远通信技术股份有限公司尽力确保开发中功能的完整性、准确性、及时性或效用，但不排除上述功能错误或遗漏的可能。除非其他有效协议另有规定，否则上海移远通信技术股份有限公司对开发中功能的使用不做任何暗示或明示的保证。在适用法律允许的最大范围内，上海移远通信技术股份有限公司不对任何因使用开发中功能而遭受的损失或损害承担责任，无论此类损失或损害是否可以预见。

保密义务

除非上海移远通信技术股份有限公司特别授权，否则我司所提供文档和信息的接收方须对接收的文档和信息保密，不得将其用于除本项目的实施与开展以外的任何其他目的。未经上海移远通信技术股份有限公司书面同意，不得获取、使用或向第三方泄露我司所提供的文档和信息。对于任何违反保密义务、未经授权使用或以其他非法形式恶意使用所述文档和信息的违法侵权行为，上海移远通信技术股份有限公司有权追究法律责任。

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。

文档历史

修订记录

版本	日期	作者	变更表述
-	2020-11-24	KingKa WU	文档创建
1.0.0	2021-02-02	KingKa WU/ Chic YE/ David TANG	临时版本： 1. 增加客户常见问题解决方案； 2. 增加章节： （1） QuecPython 官网、支持群地址 （2） QuecPython 测试问题 3. 丰富之前章节： （1） QuecPython 驱动安装失败问题解决 （2） QuecPython 救砖处理 4. QuecPython 其他常见问题；

目录

表格索引

图片索引

1 QuecPython 官网和官方交流群

移远通信 QuecPython 官方网站：http://python.quectel.com/。

移远通信 QuecPython 官方 QQ 开发交流群：445121768。

2 QuecPython 相关准备

2.1. 开发板

目前推荐的开发板：

（1） QuecPython EC100Y-CN 开发板；

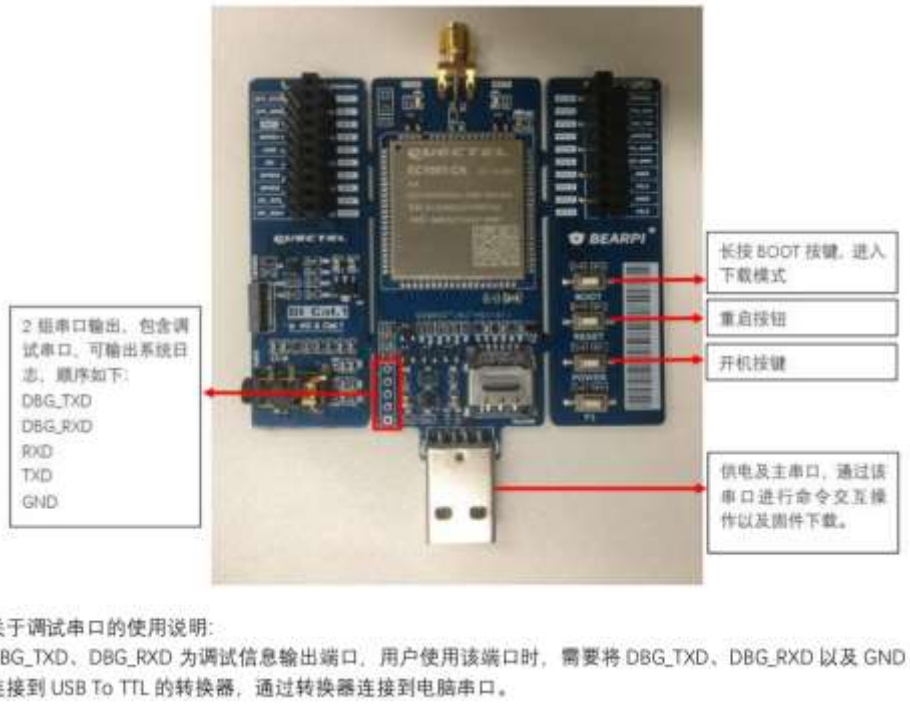


图 1：QuecPython EC100Y-CN & EC600S-CN 开发板

（2） EC600S_QuecPython_EVB_V1.x 开发板

图 2：EC600S_QuecPython_EVB_V1.x 开发板

2.2. 驱动下载安装

使用开发板前，需要在电脑上安装 USB 驱动。成功安装后，电脑方可识别开发板。

驱动程序名称	Quectel_ASR_Series_UMTS&LTE_Windows_USB_Driver_Customer
驱动程序下载地址	https://python.quectel.com/download.html

步骤一：开发者下载 USB 驱动程序的压缩包后，完整解压该压缩包到任意目录，双击运行 *setup.exe*。

图 3：运行 setup.exe

步骤二：按照提示，点击“**Install**”即可。

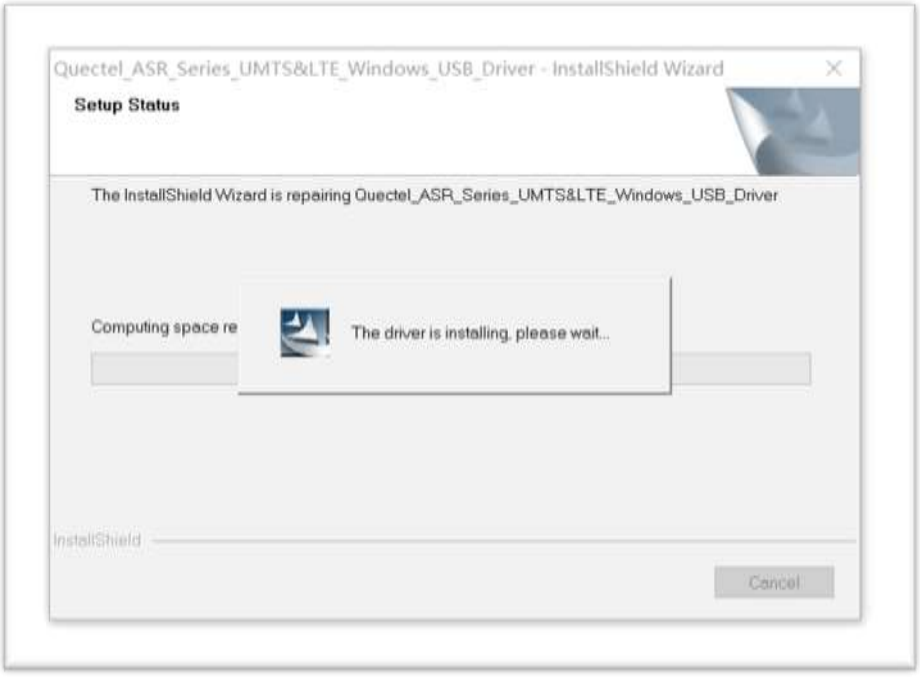


图 4：驱动正在安装

步骤三：安装成功后，点击“**Finish**”结束。

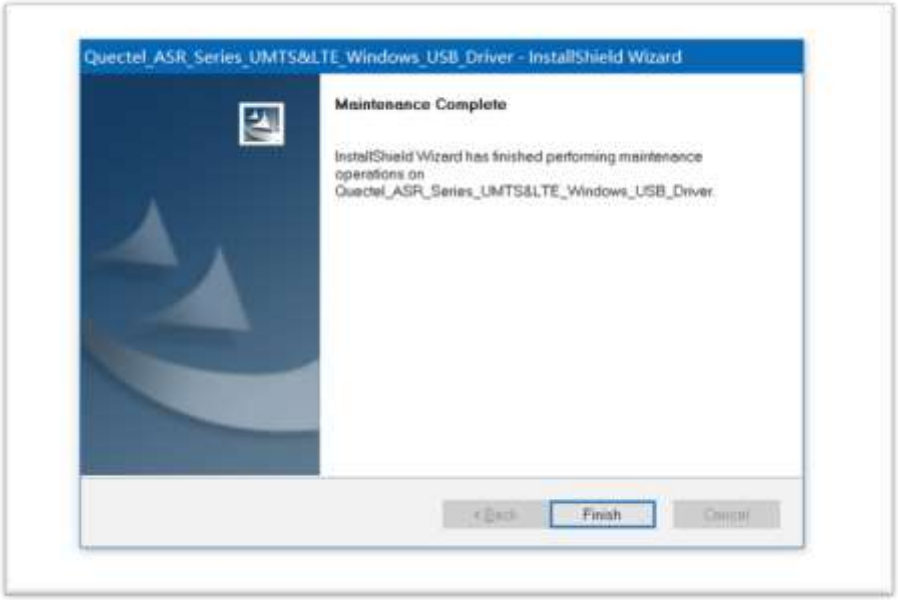


图 5：驱动安装成功

步骤四：至此 USB 驱动安装结束。模块开机后，可在“设备管理器”中的查看端口。

若模块中烧录的固件为 Python 版本，模块开机后，“设备管理器”中的端口显示如下：

图 6：Python 固件版本端口显示

若模块中烧录的固件非 Python 版本，模块开机后，“设备管理器”中的端口显示如下：

图 7：非 Python 固件版本端口显示

备注

- USB 驱动安装失败问题请参考***第 6.1 章***。
- 安装前，请备份重要文件并保存工作进度，以免发生意外情况导致文件丢失。
- USB 驱动安装成功后，无需重启电脑。
- 如需修复或者卸载驱动程序，再次运行该驱动安装程序，选择“**修复**”或“**卸载**”即可。

2.3. 固件下载安装

- 为了对用户 Python 应用程序做交互适配，需要先下载 QuecPython 固件到开发板中，下载完成后才可以对 Python 应用程序做底层适配处理。固件下载网址：https://python.quectel.com/download.html。
- 支持通过 QPYcom 工具烧 Python 版本固件，该工具不支持烧录非 Python 版本固件。
- 如果模块中当前固件为 Python 版本（固件版本号以“PY”结尾，如“EC100YCNAAR01A01M16_OCPU_PY”），则直接跳过此步骤；如果非 Python 版本（固件版本号不以“PY”结尾），根据项目需求，请至 QuecPython 官网分别下载对应的 EC100Y-CN，EC600S-CN 固件。

步骤一：确认当前模块中烧录的固件版本。在 QCOM 中发送命令 **AT+GMR** 进行查看，如下图所示：

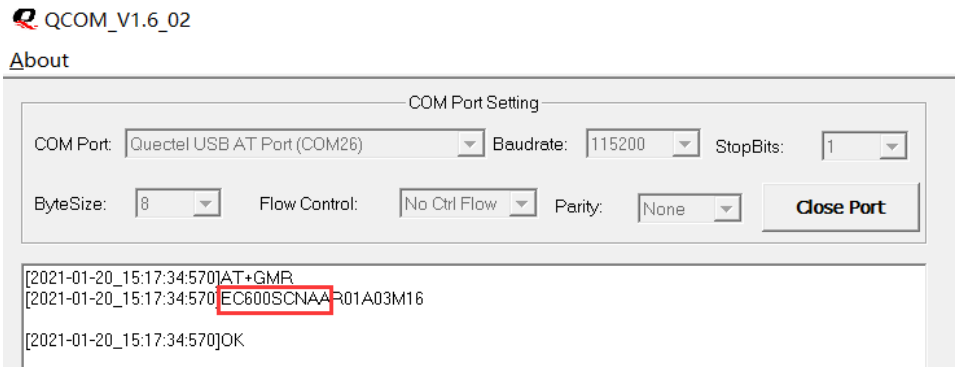


图 8：使用 QCOM 查看当前固件版本

步骤二：请勿打开串口，直接创建项目（对于非 Python 固件的模块一定不要打开串口）。

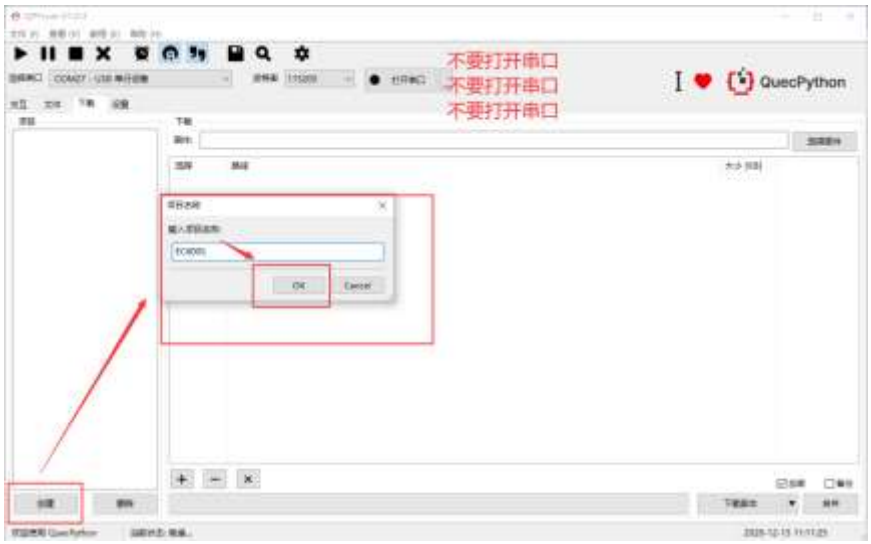


图 9：创建项目

步骤三：点击“**选择固件**”。

图 10：选择固件界面

备注

1. 该步骤选择的固件包为压缩包。
2. 下载固件前需确认固件包里无额外的压缩包

步骤四：点击“**下载固件**”。

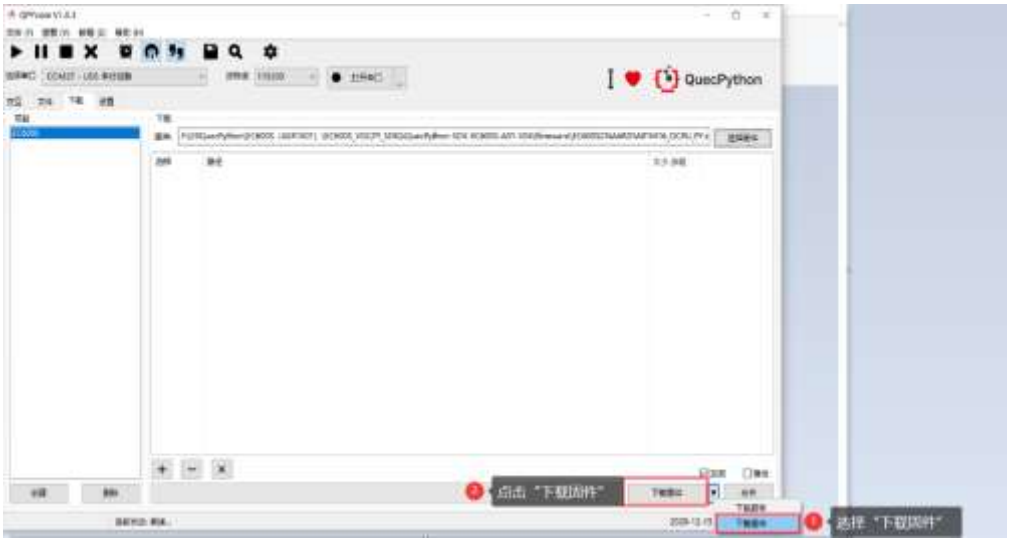


图 11：下载固件界面

等待 20 秒左右会出现下载进度条。

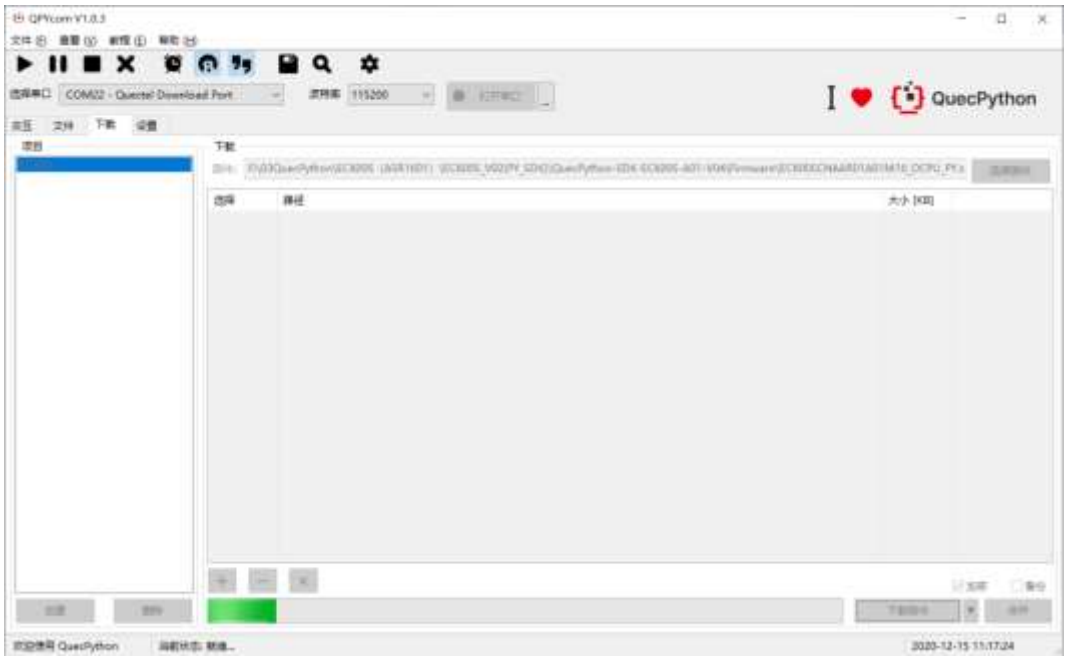


图 12：固件下载进度

步骤五：“下载结束”出现在下方界面即表示下载成功。

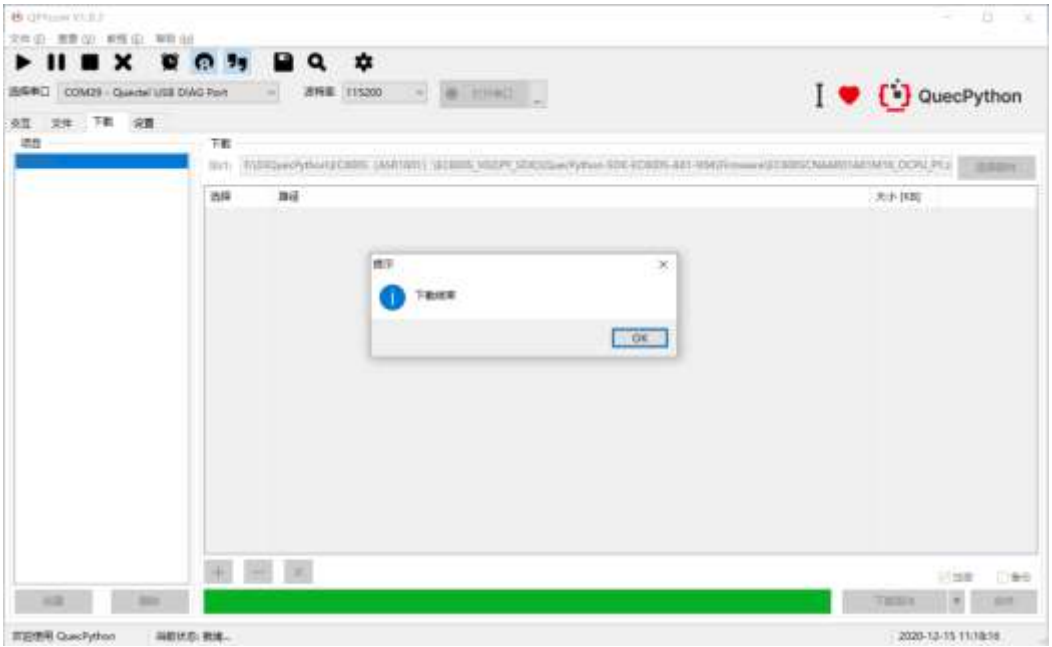


图 13：下载成功界面

备注

1. QuecPython 固件安装失败问题请参考***第 3 章***。
2. 安装前，请备份重要文件并保存工作进度，以免发生意外情况导致文件丢失。
3. 安装成功后，无需重启电脑。

2.4. 开发板脚本下载

2.4.1. 脚本下载建议

1. 为方便脚本下载，QPYcom 已经做了脚本拖拽下载的功能以供使用，详细信息可参考《Quectel QuecPython_QPYcom 工具使用说明》。
2. 如需一次性下载所有脚本，并清除原来的脚本（关于脚本清除的内容请参考***第 2.4.2 章***），可以按照下面的脚本下载步骤进行操作。

2.4.2. 关于脚本清除的解释

对于根目录除了 Json 都会删除，而对于子文件夹（用户自行创建的文件夹），脚本下载不会影响这个文件夹。

2.4.3. 脚本下载步骤

步骤一：选择“USB 串行设备”后，依次点击**“打开串口”**和**“下载”**按钮，创建项目（项目名称可任意命名）。

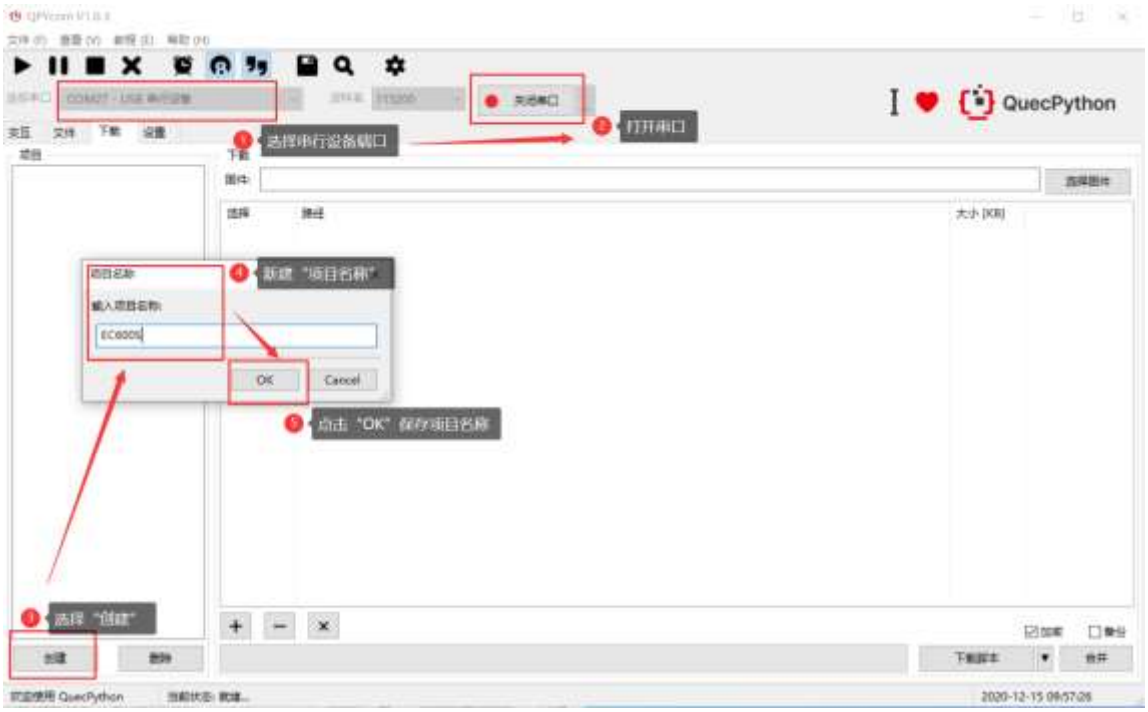


图 14：创建项目

步骤二：点击“**选择固件**”（脚本下载暂时是需要选择固件，但固件不会被下载）。

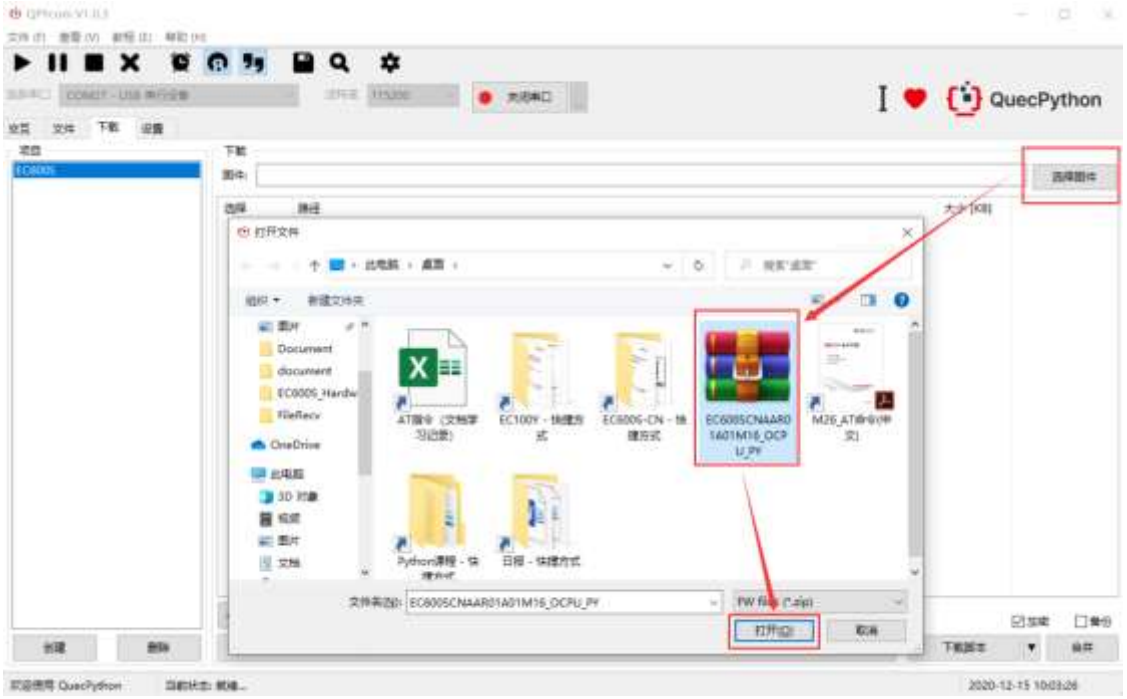


图 15：选择固件

步骤三：选择需要下载的脚本。

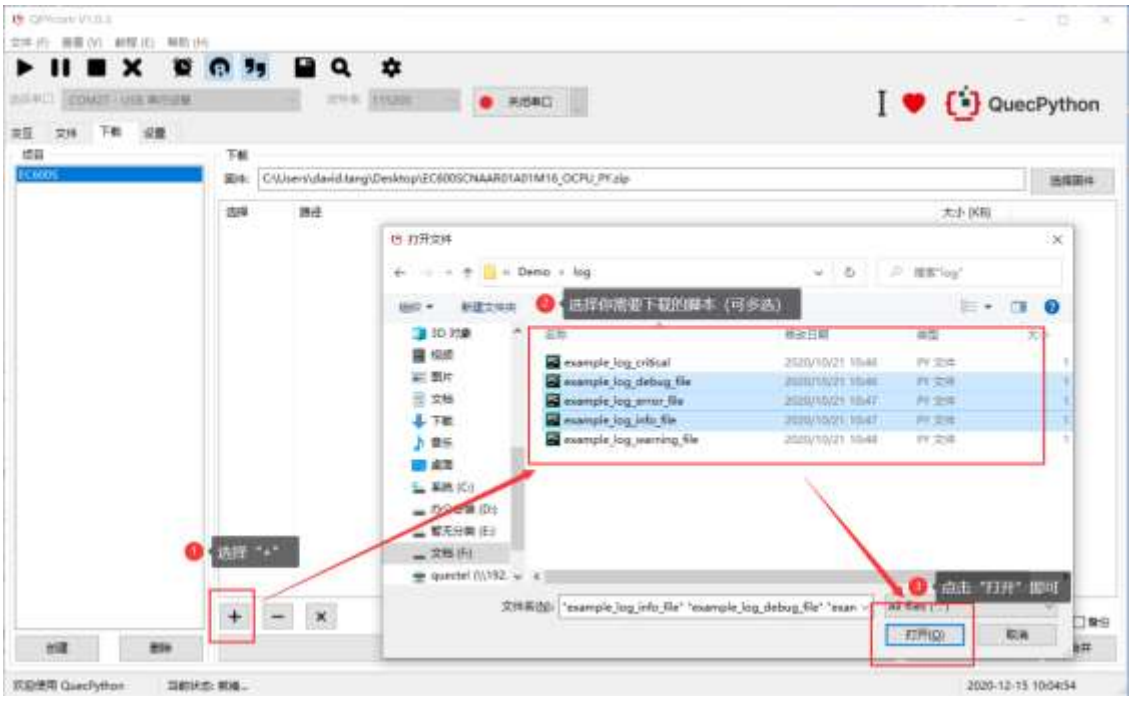


图 16：选择脚本

步骤四：点击“**下载脚本**”，等待脚本下载完成。

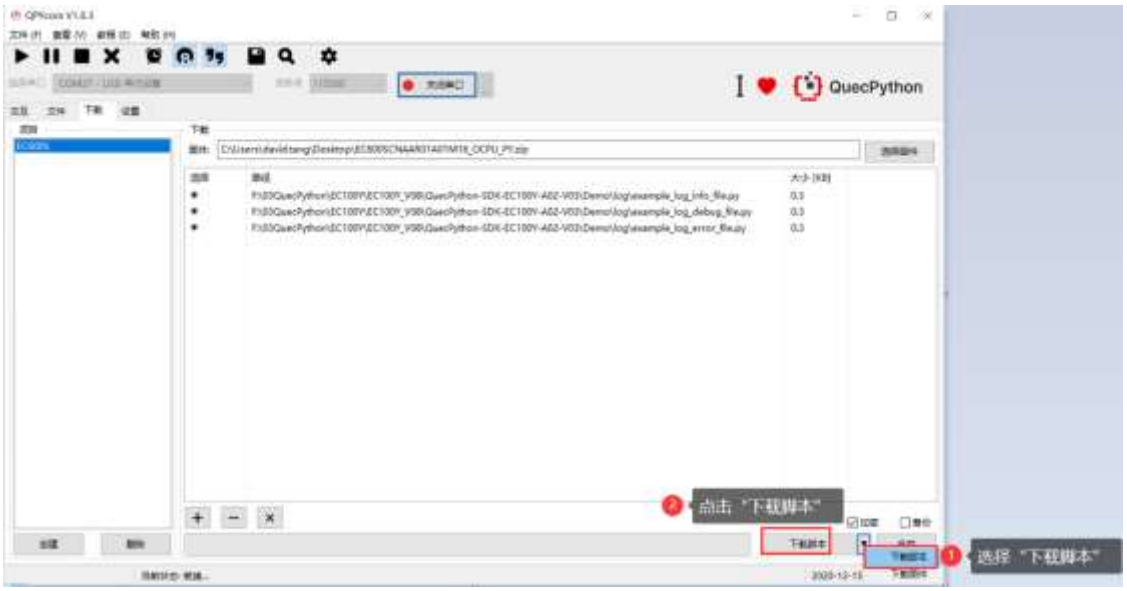


图 17：下载脚本

步骤五：完成界面展示。

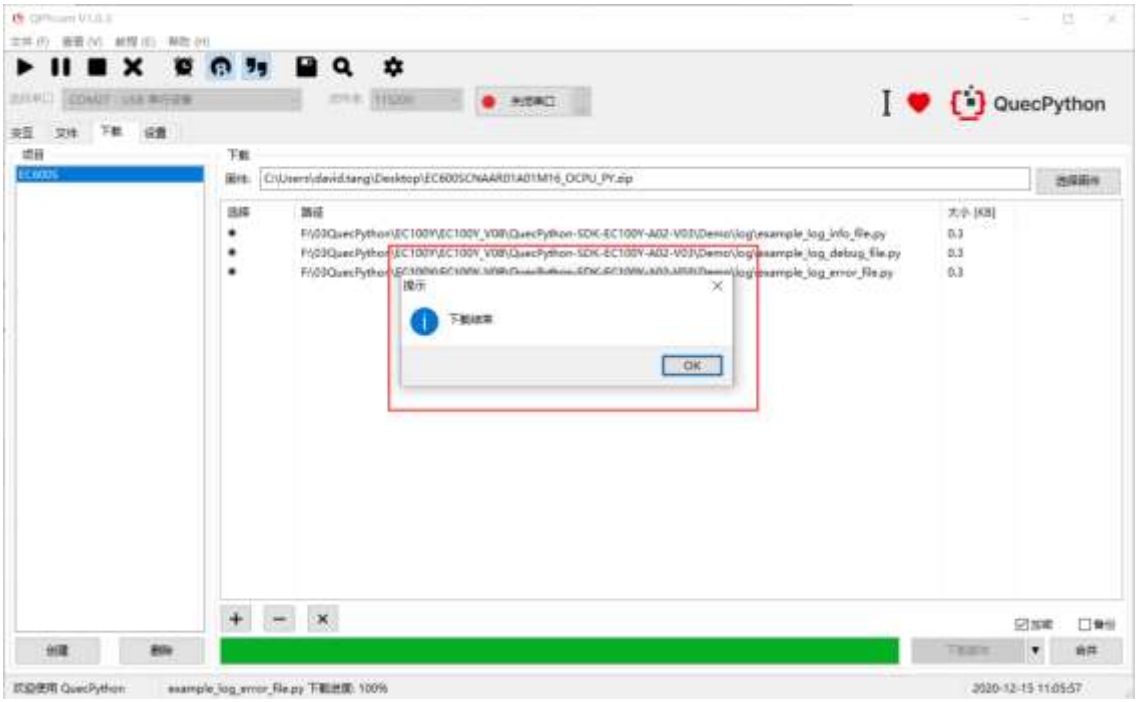


图 18：下载结束界面

3 QuecPython 救砖处理

当使用 QPYcom 工具下载固件失败后，无需担心板子成砖，可使用以下两种方法成功下载固件。

3.1. 方法一

步骤一：打开电脑任务管理器（可按快捷键 **Ctrl+Alt+Delete** 打开），找到 *QPYcom.exe*，强制结束后台任务。

步骤二：对于开发板，按住板子上的“reset”按键，重启板子；对于裸模块，可直接断电再上电，然后拉低 POWKEY 开机。

步骤三：查看“设备管理器”的端口列表是否显示正常，参考***第 2.2 章***中的步骤四。若端口显示正常请进行步骤四的操作；若端口显示异常，则参考***第 3.2 章***。

步骤四：如果正常显示出串口则重新打开 QPYcom.exe 工具，参考《Quectel QuecPython_QPYcom 工具使用说明》重新下载固件。

3.2. 方法二

步骤一：下载 QFlash 工具压缩包并解压后。

步骤二：双击运行该软件，例如：*QFlash_V5.0.exe*。

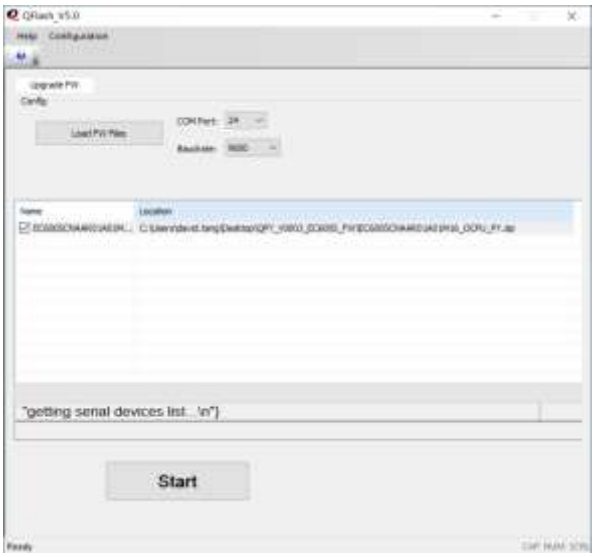


图 19：QFlash 工具

步骤三：点击“**Load FW Files**”，选择固件升级包。

备注

1. 该步骤选择的固件包为压缩包。
2. 下载固件前需确认固件包里无额外的压缩包

步骤四：对于 EC100Y-CN 和 EC600S-CN 模块，都无需选择“COM port”和“Baudrate”，换而言之，即使没有正常端口，也可以使用 QFlash 来进行固件烧录。

步骤五：上电前短接 VDD_EXT 和 USB_BOOT，然后上电，同时点击 QFlash 工具中的“**Start**”按钮进行固件烧录。

短接 VDD_EXT 和 USB_BOOT 重新上电后，“设备管理器”中的端口显示如下图所示（禁用其他端口，仅出现下载口）。

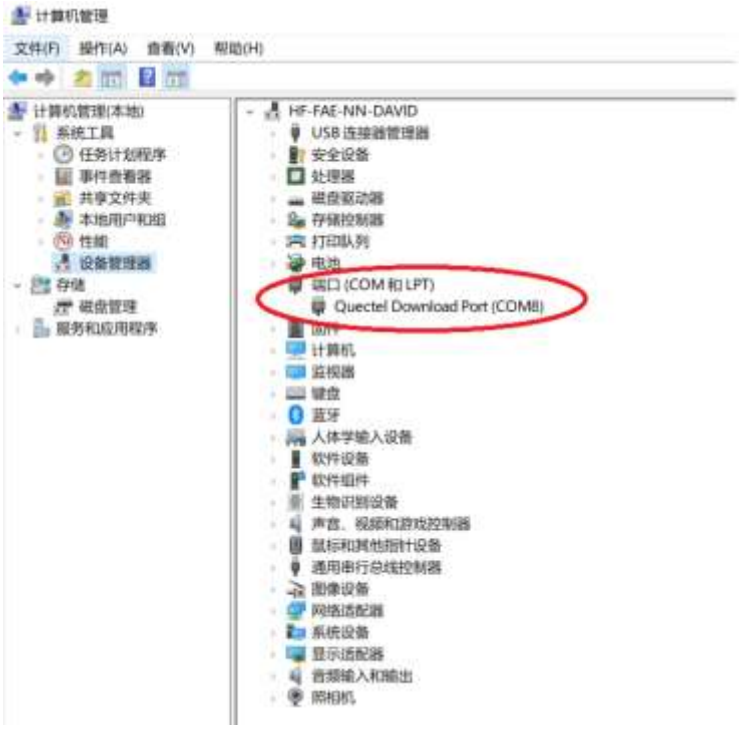


图 20：端口显示

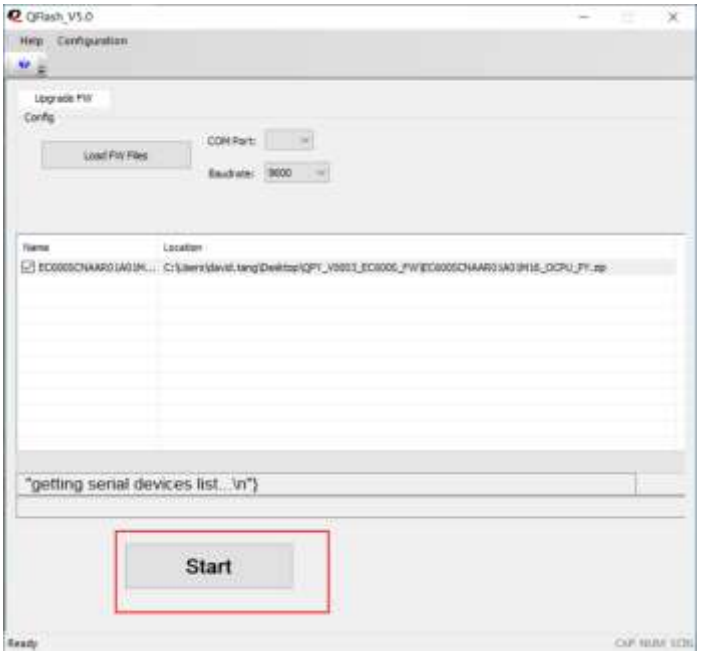


图 21：QFlash 界面

备注

由于 QFlash 检测下载口有时间限制，可以考虑在操作**步骤二**前短接 VDD_EXT 和 USB_BOOT，然后操作**步骤五**时，直接上电即可。否则，需快速操作**步骤五**。

步骤六：可在出现如下图所示提示，表明固件烧录完成。

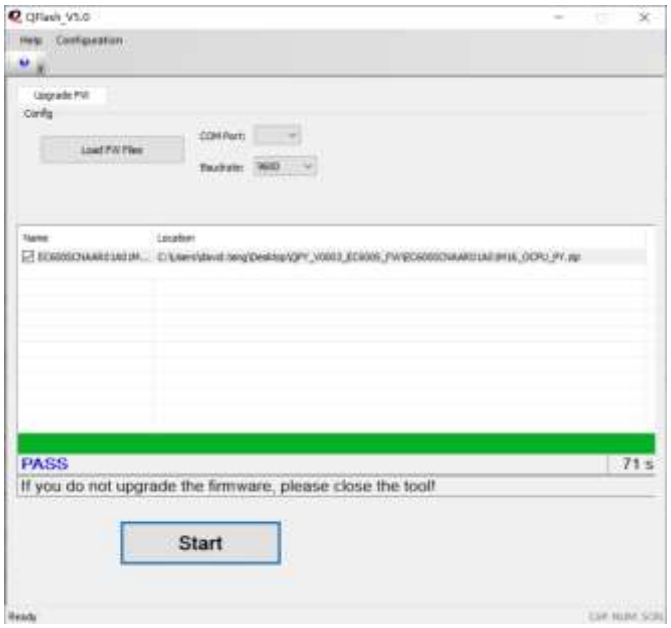


图 22：固件烧录完成界面

步骤七：可在 QCOM 中执行 **AT+GMR** 进一步验证固件是否成功烧录。

图 23：QCOM 查询界面

备注

若 QFlash 烧录固件失败，请关闭并重新启动 QFlash（可以通过“Ctrl+Alt+Delete”打开“任务管理器”，彻底杀死 QFlash 进程），然后重新回到步骤二进行固件烧录操作。

4 QuecPython 代码编写及下载调试

4.1. QuecPython API 类库文档

QuecPython API 类库是指提供给用户使用的模块接口。用户通过这些接口可以知道目前支持和不支持的模块功能，同时可以根据该 API 类库中模块的不同组合开发出自己想要的产品及功能。

如需获取 QuecPython API 类库文档，可以访问移远通信 QuecPython 官方网站：https://python.quectel.com/wiki/。

图 24：QuecPython API 类库

通过 QuecPython API 类库文档可以看出目前主要支持的有 MicroPython 标准库、QuecPython 类库及第三方库。其中 QuecPython 类库主要包含了数据拨号、基站定位、网络相关、FOTA 升级、音频播放及其他 BSP 驱动接口，第三方库主要包含阿里云、腾讯云、MQTT、HTTP 等类库。

备注

在 QuecPython 官网上有大量的教学文档和入门视频，可供用户参考使用。

4.2. 编写 Python 源程序代码

在开发产品过程中，可以通过 Pycharm IDE 工具或 notepad++进行源程序编写。

移远通信 QuecPython 官网提供了多个教学文档，每个教学文档中都有源代码（QuecPython 驱动子程序和 QuecPython 小实验），欢迎在官网下载（https://python.quectel.com/wiki/）。

图 25：QuecPython 官网教学文档

以第一个“hello python”代码为例，打开 notepad++编辑器后输入以下代码，将文件保存并命名为 *hello_world.py*，即完成编写第一个 Python 源程序。

```
hello_python.py
1
2  print("hello python")
```

4.3. QPYcom 工具脚本下载调试

按照*第 4.2 章*的步骤，已编写好第一个 Python 源程序代码，接下来使用 QPYcom 工具下载脚本到模块中运行。

步骤一：首先打开 QPYcom 工具，进入到下载界面，并依次按照*第 2.4.3 章*所述步骤进行脚本下载。

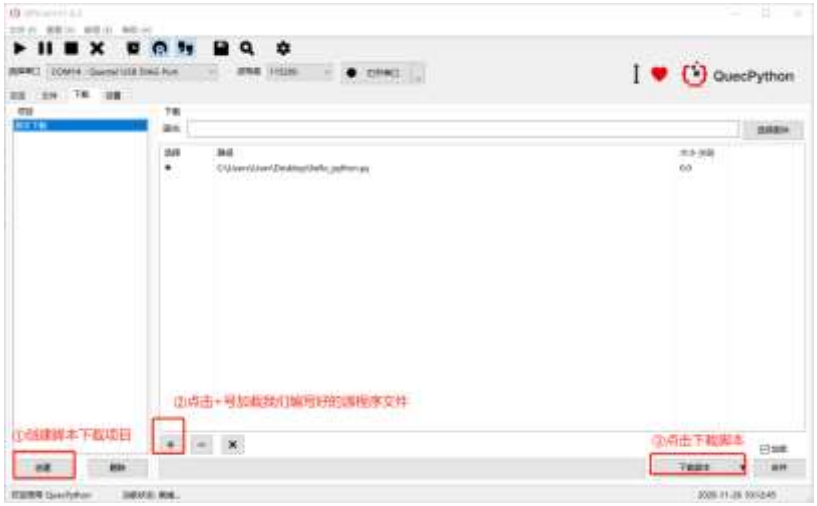


图 26：QPYcom 下载界面

步骤二：下载脚本结束。

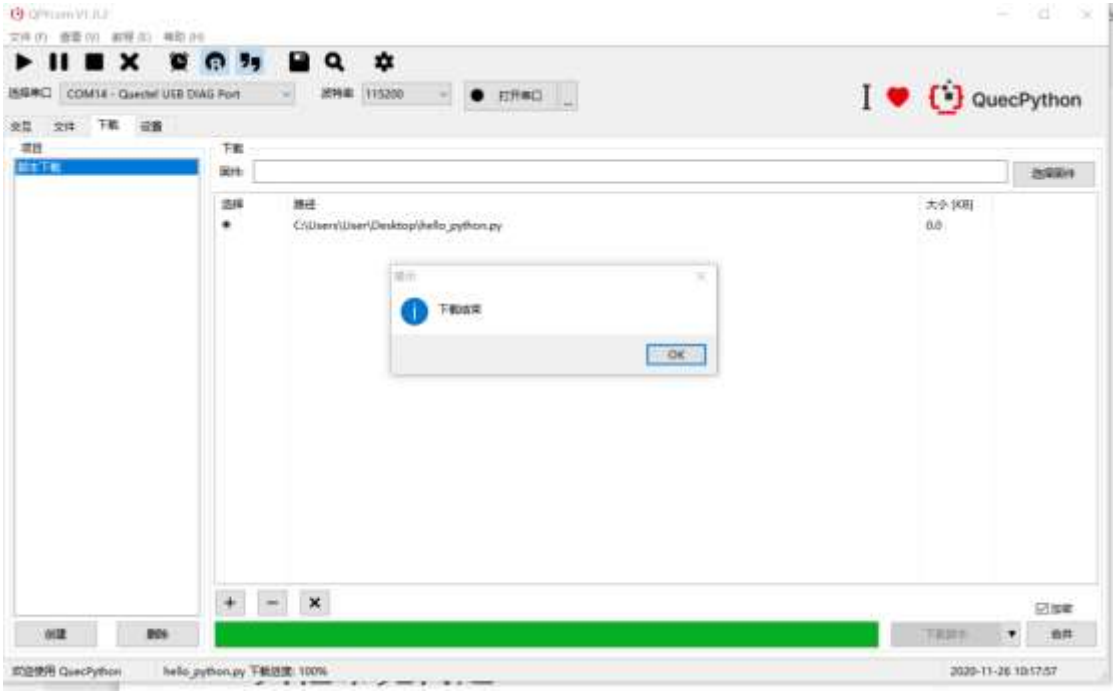


图 27：脚本下载完成

步骤三：在文件列表查看已下载的脚本，并点击运行按钮，运行脚本。

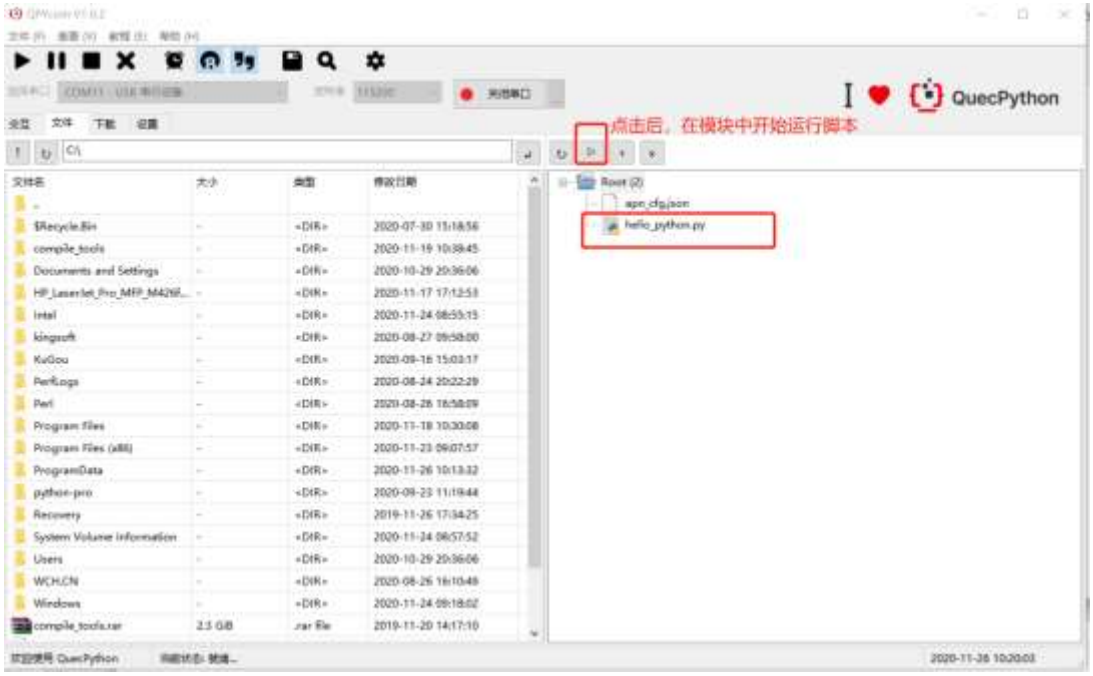


图 28：查看已下载的脚本

步骤四：运行结果如下图所示。

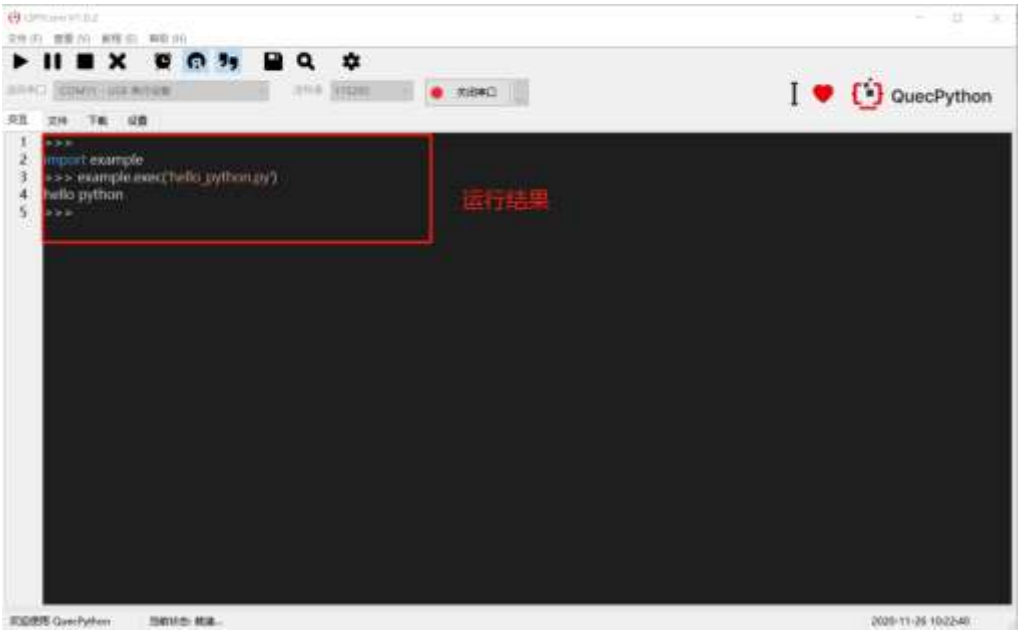


图 29：脚本运行结果

5 QuecPython 注意事项

5.1. QuecPython main.py 文件使用

5.1.1. main.py 使用建议

对于开始的调试，建议不要将程序命名为 *main.py*，可采用其他命名，例如：*start.py* 等等。如果文件代码中存在死循环，且在程序运行时无法中断，会导致程序阻塞，暂时只能通过重刷固件解决。

5.1.2. 常见问题

1. 现象 1：上传命名为 main.py 的文件至模块后无法任何执行指令（包括上传文件等），类似现象如下所示。

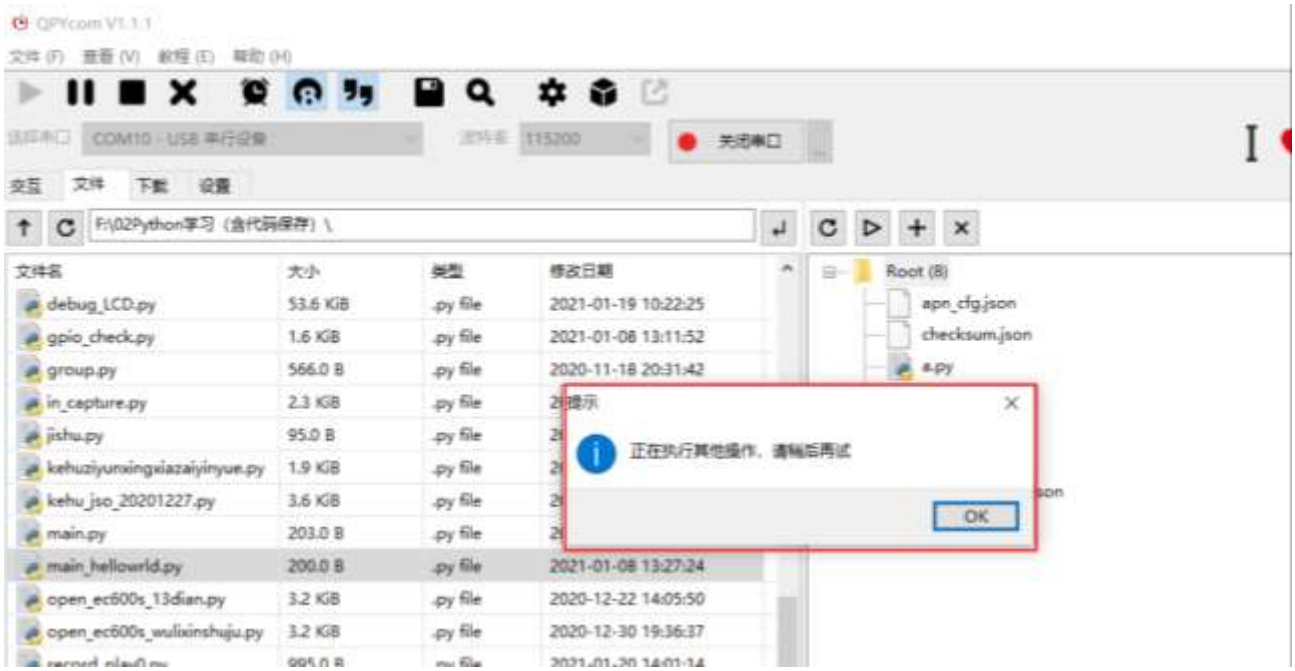


图 30：上传 PY 文件失败

- 原因

模块在开机后会自动寻找运行文件名为 *main.py* 的脚本文件，如果 *main.py* 中存在 *while*，*for(,,)*等循环语句，会导致程序阻塞，串口被占用，无法进行其他操作。

- **解决方法**

重新烧录固件。

若使用 *main.py* 程序的需求非常强烈，同时又需要使用死循环，必须在死循环里面加一个可中断的条件，示例如下：在一个线程里面加入了死循环，同时在这个死循环里面加入了中断退出的条件，当一直输出时，可通过 GPIO2 对应的按键中断输出。

```
import log
import _thread
from machine import Pin
import utime
log.basicConfig(level=log.NOTSET)
KEY_log = log.getLogger("KEY")
gpio2 = Pin(Pin.GPIO2, Pin.IN, Pin.PULL_DISABLE, 0)
def in_capture():
    KEY_log.debug("in_capture start!")
    while True:
        KEY_log.info("1111")
        utime.sleep(1)
        if gpio2.read() == 0:
            KEY_log.info("in_capture thread end")
            break
        else:
            pass
if __name__ == "__main__":
    _thread.start_new_thread(in_capture, ())
```

2. 现象 2：手动运行 main.py 程序，通过 QPYcom 连接“USB 串行设备”有 print 输出和 LOG 打印信息，但是自运行 main.py 程序后，在 QPYcom 的交互界面，无任何信息打印。

- **解决方法**

完成以下软硬件配置，加入串口打印。

（1） 软件代码参考如下，串口打印需要导入 UART 模块，按照 UART 的 API 库来书写输出代码。

```
from machine import UART   #导入 UART 模块
uart = UART(UART.UART2, 115200, 8, 0, 1, 0)   #配置成 UART2 输出（硬件的连接）
count = 50
while count:
    uart.write('main_py_UART_msg:{}\r\n'.format(count))
    count -= 1
```

（2） 硬件连接可参考：

对于代码中提到的硬件连接（UART2），是使用 type-c 给模块供电，UART 与 TTL 转 USB 模块的连接如下表，TTL 转 USB 模块直接插 PC 上：

模块 UART_pin 脚	TTL 转 USB 模块
RX1	Tx
TX1	Rx
GND	GND

使用 QCOM 连接“TTL 转 USB 模块”对应的串口，如下图所示

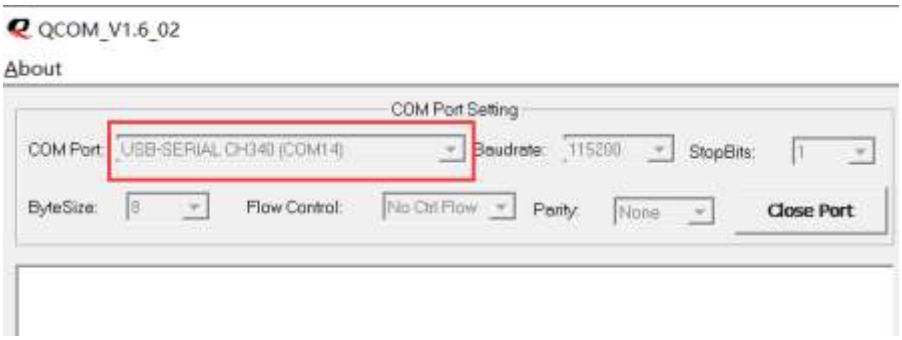


图 31：使用 QCOM 连接“TTL 转 USB 模块”对应串口

完成上述的软硬件配置后，在自运行 *main.py* 程序时，在 QCOM 即可输出对应打印信息，如下图：

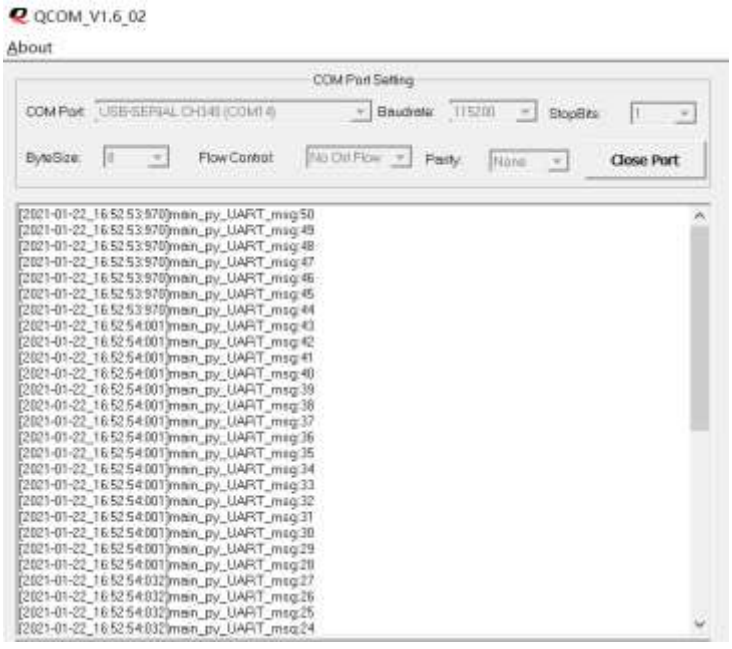


图 32：自运行 main.py 程序

5.2. 串口的收发数据测试流程

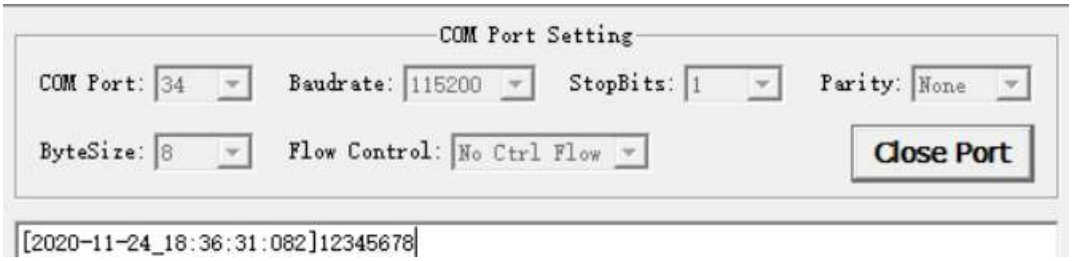
- 环境搭建（仅用于测试 API 对串口的封装）：

（1）硬件连接：USB 给 EC600S_QuecPython_EVB_V1.x 供电，Main 串口的 TX1 和 RX1 对应接到电平转换芯片的 3.3V 端（经电平转换成 5V，通过一个电平转换器件），串口工具 GND 接地。

（2）软件方面：一端使用 QPYcom 连接 EC600S_QuecPython_EVB_V1.x 的“USB 串行设备”口（用于调试时发送 QuecPython 指令）；另一端使用 QCOM 连接“电平转换器件”对应的端口（用于接收 EC600S_QuecPython_EVB_V1.x 发送的数据（测试模块的 write 功能），以及发送数据给模块（测试模块的 read 功能）。

- 发数据（write）

```
>>> from machine import UART
>>> uart = UART(UART.UART2,115200,8,0,1,0)                ////////////////////  定义发数据的端口，必须要严格按照硬件的接线来配置
>>> uart.write("12345678")        ////////////// write 数据后，可以通过 QCOM 看到数据（如下截图）
8       //////////////“8”表示发送的字节数
```



- 收数据（read）

```
>>> from machine import UART
>>> uart = UART(UART.UART2,115200,8,0,1,0)          ////////////////////  定义接收数据的端口，必须要严格按照硬件的接线来配置
>>>  uart.any()          ////////////////  显示缓存的数据（只有在 QCOM 发送数据，这里才会显示未读的数据）
32
>>> msglen = uart.any()                    ////////////////////  将缓存的数据字节长度数赋值给 msglen
>>> msg = uart.read(msglen)                ////////////////////  读取缓存字节数对应的数据
>>> utf8_mmmm = msg.decode()               ////////////////////  将 byte 类型的数据转换成 unicode 类型
>>> print(utf8_mmmm)                       ////////////// 以 unicode 类型输出数据
555555
555555
555555
555555
>>>
```

5.3. EC100Y-CN 休眠_低功耗测试

5.3.1. 休眠条件

满足以下条件，EC100Y-CN 模块方可进入休眠。

（1） EC100Y-CN 模块最多支持 32 把唤醒锁，如果需要进入休眠，必须释放所有的锁。（具体命令参考 API 库：pm -低功耗）

（2） 配置自动进入休眠（具体命令参考 API 库：pm -低功耗）

5.3.2. 确认休眠状态

（1）从 AT 口，发送 **AT+LOG=80,1**，然后在 DEBUG 口查看消息。

主要查看前四个参数：

- 休眠总开关
- 允许进入休眠开关
- USB 状态
- UART 状态

如上图所示，以上四个参数是“1，1，0，0”基本可以表明模块已经进入了休眠。后面的参数是唤醒寄存器的值，通过该值可以具体查看哪些唤醒源没有被清除。

（2）进入休眠状态测试：检查实际电流值。

5.3.3. 唤醒条件

（1）外部引脚中断唤醒（需要配置和使能，不是普通的 GPIO 中断），对于中断唤醒如何操作可参考 ***第 5.3.4 章***。

（2）电话、短信、网络数据唤醒。

5.3.4. EC100Y-CN 支持 PCM 以及中断唤醒

如何进入睡眠可以参考 PM（低功耗）的 API 库，对于中断唤醒需要参考“ExtInt”的 API 库，操作配置流程说明：

（1） 如果想进入睡清除所有的锁，此时可以通过“pm.autosleep(sleep_flag)”自动进入睡眠。

（2） 但是此时来中断时，是通过 ExtInt 的 API 库，配置进入 callback，在回调函数中，进行判断是否加

锁，如果是可唤醒模块的中断，则对其进行加锁，来唤醒模块（即在睡眠的时候，代码仍在运行中）

5.4. EC600S-CN 裸模块开机烧录测试

表 1：EC600S-CN 硬件连接

模块端	USB
USB_DP（PIN26）	绿色－USB 数据线（正）
USB_DM（PIN27）	白色－USB 数据线（负）
USB_VBUS（PIN28）	红色－USB 电源
GND（PIN30）	黑色－地线
模块端	**电源端**
VBAT_BB（PIN29）	电源的正极
VBAT_RF（PIN36）	
VBAT_RF（PIN37）	
GND（PIN38）	电源的负极

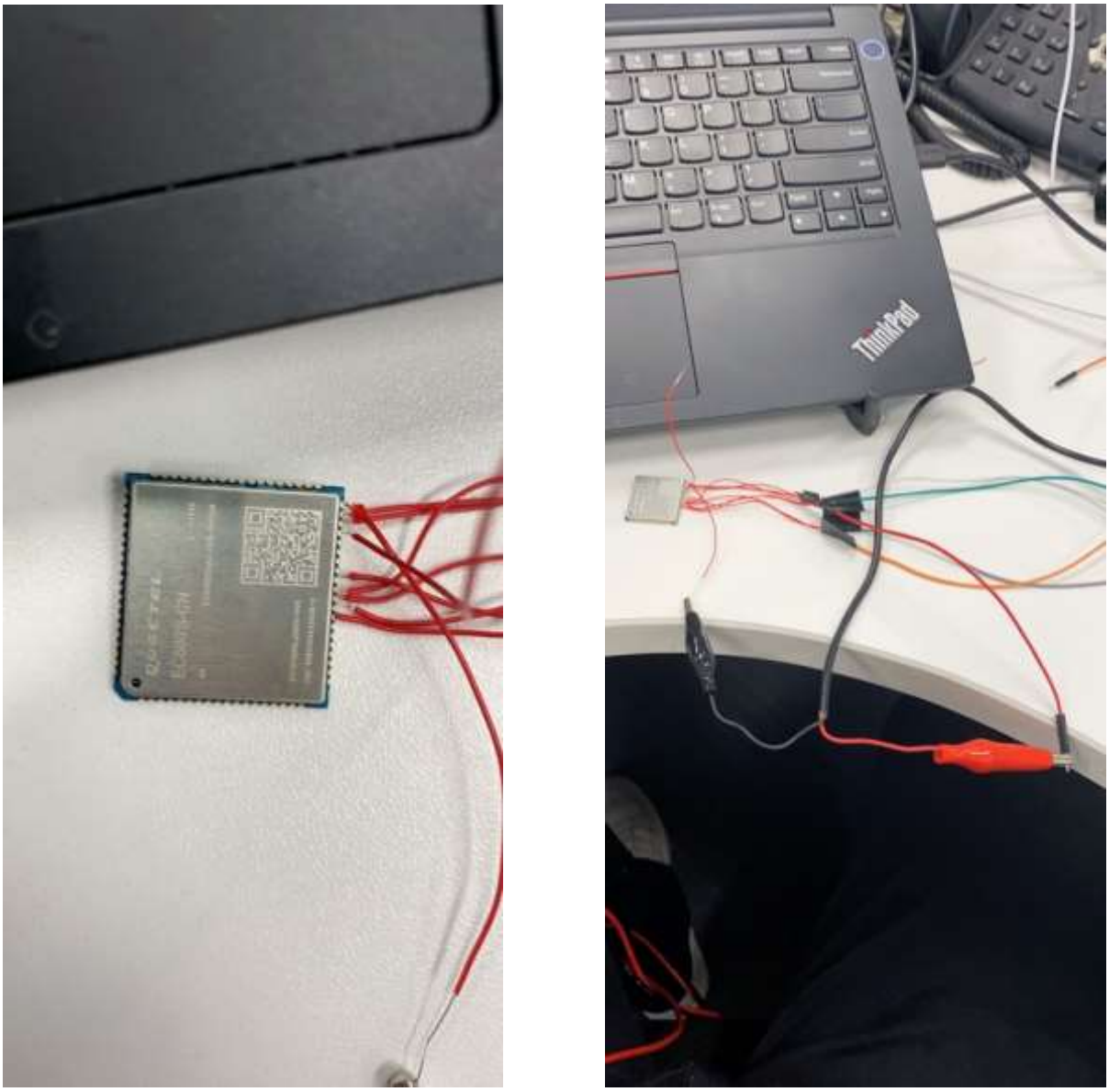


图 33：硬件连接实物图

- 若模块中烧录的固件非 Python 版本，拉低 POWERKEY 开机后，“设备管理器”中的端口显示如下：

图 34：非 QuecPython 固件版本端口显示

- 烧录 Python 版本固件，重启模块后，“设备管理器”中的端口显示如下：

图 35：烧录 Python 固件版本后端口显示

图 36：烧录前后版本号对比

5.5. QuecPython MQTT 连接

5.5.1. Umqtt 使用

umqtt 模块提供创建 MQTT 客户端的发布订阅功能。该模块可以向服务端发布或者订阅消息，向服务器发送 ping 包，检测保持连通性，与服务器建立或者断开连接。需要注意的是 *MQTTClient.check_msg()* 和 *MQTTClient.wait_msg()*方法中推荐使用 *wait_msg* 的方法。

5.5.2. 使用 MQTT 连接阿里云、腾讯云等

使用MQTT连接阿里云、腾讯云的详细步骤可以参考官网上wiki社区（https://python.quectel.com/wiki/）的 QuecPython 云服务。

5.5.3. MQTT 连接异常处理

5.5.3.1. 云服务运行 demo 后出现未订阅等提示

现象：类似如下截图。

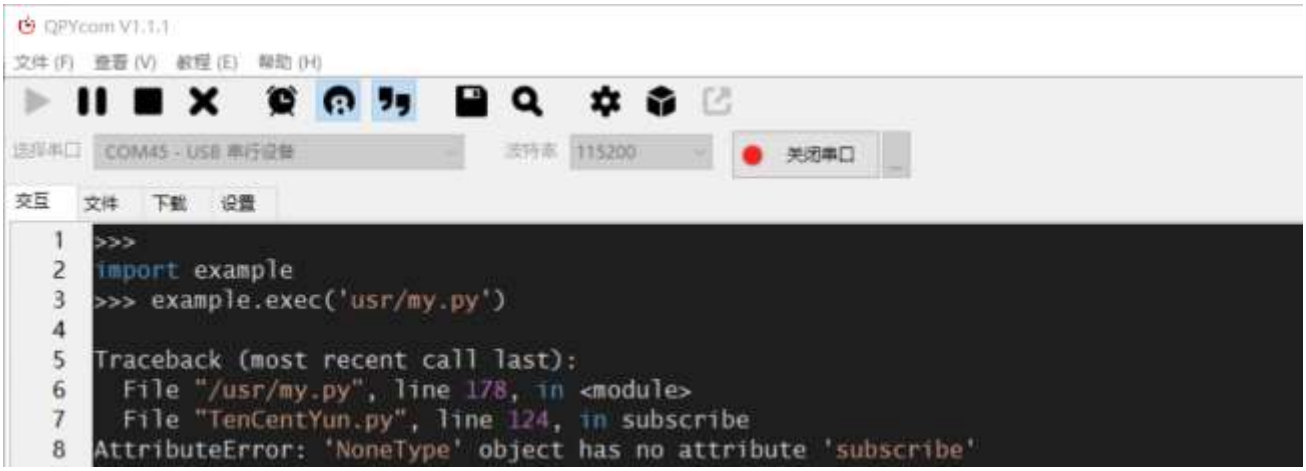


图 37：云服务运行 demo 出现未订阅提示

1. 原因分析

- 注网失败；
- 用户订阅与发布的主题，在云服务端未创建。

2. 解决方案

（1）对于注网失败，解决步骤如下：

在 QCOM 中执行查询命令“**AT+CSQ; +CREG?; +CEREG?; +CGREG?; +CPIN?;+COPS?**”，如下截图，类似下面情况说明网络状态是正常的。

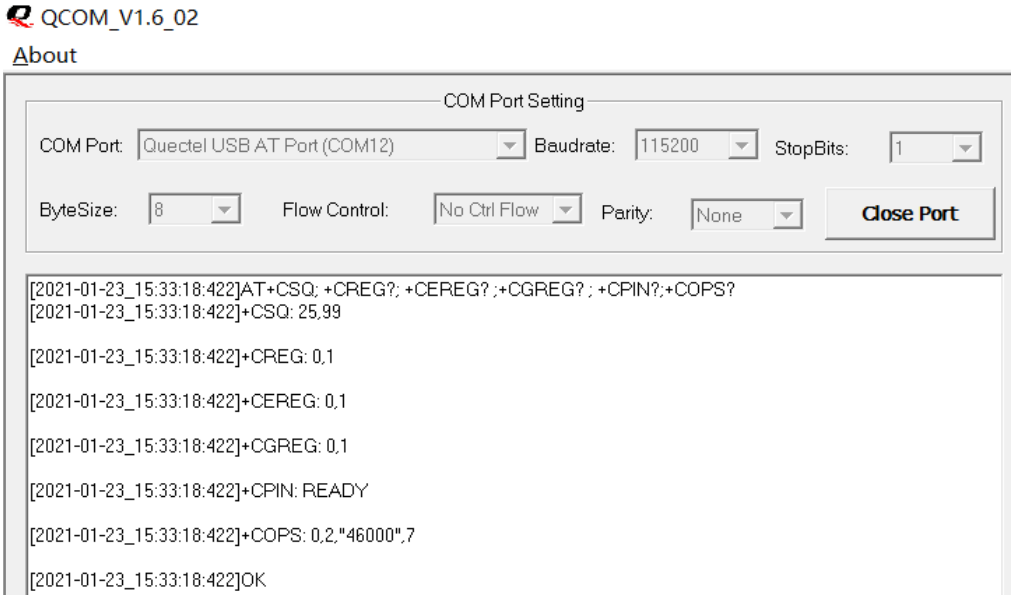


图 38：查询网络状态

若返回异常：

- 对于 **AT+CPIN?**的异常返回，检查 SIM 卡座硬件问题；
- 对于 **AT+CREG?**、**AT+CGREG?**、**AT+CEREG?**的异常，建议检查手机卡是否存在欠费等情况；
- 对于 **AT+CSQ?**的异常返回，检查周围环境，是否存在信号干扰，建议室外测试，或者室内使用天线。

（2）对于“订阅与发布的主题，在云服务端没有创建”，建议检查以下两点：

- 检查订阅的主题，在云服务上对应的主题是不是也是可订阅的状态；
- 检查发布的主题，在云服务上对应的主题是不是也是可发布的状态。

5.5.3.2. MQTT 连接一段时间异常断开

导致原因：MQTT 服务端会有心跳检测机制，一段时间内设备与云端没有通信活动会主动断开连接。

尝试解决方向：连接断开是依据配置 MQTT 时的超时值 keepalive，在超出活动时间后会主动断开连接，用户可以使用定时器在 keepalive 活动时间超出前主动向云端发送 ping 包，服务端返回的数据包无需处理。

6 QuecPython 其他常见问题

6.1. USB 驱动安装失败

若驱动已安装，但是 USB 枚举失败，解决方法如下：

（1） 在官网下载最新的对应的固件。

（2） 在“控制面板”——“卸载程序”中，查看驱动是否安装成功，安装成功状态如下图所示：

如果仍存在驱动安装问题，请加入移远通信 QuecPython 官方 QQ 开发交流群：445121768，工程师在线答疑解惑。

6.2. 如何操作开发板

参考对应开发板的使用指导文档，下载地址：https://python.quectel.com/download.html。

6.3. EC100Y-CN 开发板与 EC100Y-CN 模块的串口位置

EC100Y-CN 开发板（小熊派）的串口位置如下图：

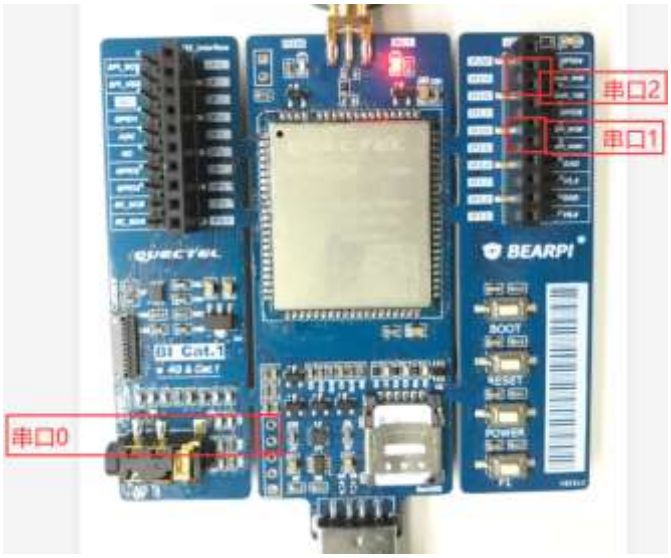


图 39：EC100Y 开发板的串口位置

EC100Y-CN 模块的串口位置可参考《Quectel_EC100Y-CN_QuecOpen_硬件设计手册_V1.0》（可在移远 QuecPython 开发交流群的群文件自行下载）。

6.4. EC600S-CN 开发板与 EC600S-CN 模块的串口位置

EC600S-CN 开发板（EC600S_QuecPython_EVB_V1.x 开发板）的串口位置如下图所示。

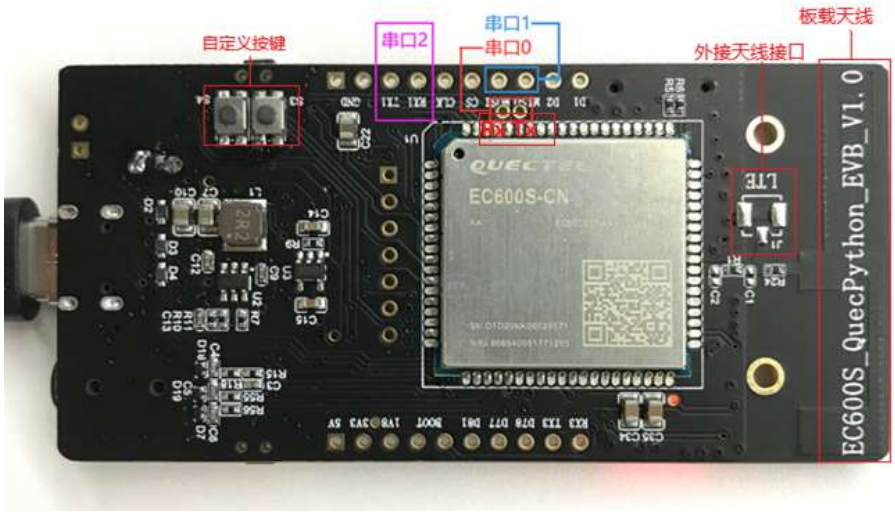


图 40：EC600S-CN 开发板的串口位置

EC600S-CN 模块的串口位置可参考《*Quectel_EC600S-CN_QuecOpen_硬件设计手册_V1.0.0_Preliminary_20200927*》（可在移远 QuecPython 开发交流群的群文件自行下载）。

6.5. QuecPython 的 GPIO 对应关系说明

步骤一：见下图 GPIO 与 PIN 脚的对应关系（左侧是 QuecPython 的 GPIO 命名，右侧是模块的 PIN 脚号）。以 QuecPython 的 GPIO6 为例，QuecPython 的 GPIO6 对应模块的 PIN15；

图 41：GPIO 与 PIN 脚的对应关系

步骤二：由步骤一已获知 QuecPython 的 GPIO6 对应模块的 PIN15，见如下 EC600SV1.1 的原理图，（无需关注黑框，只需关注红框的内容），PIN15 对应 GPIO77（GPIO77 实际上没有特别的含义，仅表示一个连接关系，也就是此处的 GPIO77 对应于下图中的 GPIO77）。

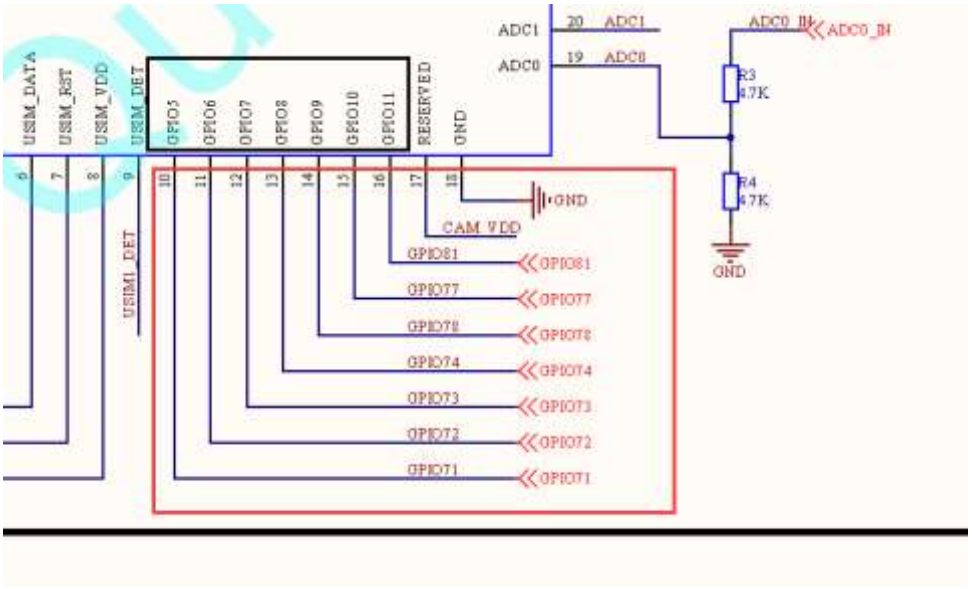


图 42：EC600SV1.1 原理图

步骤三：如下图所示，GPIO77 对应开发版本引出的 GPIO77（开发板的 J6 处）。

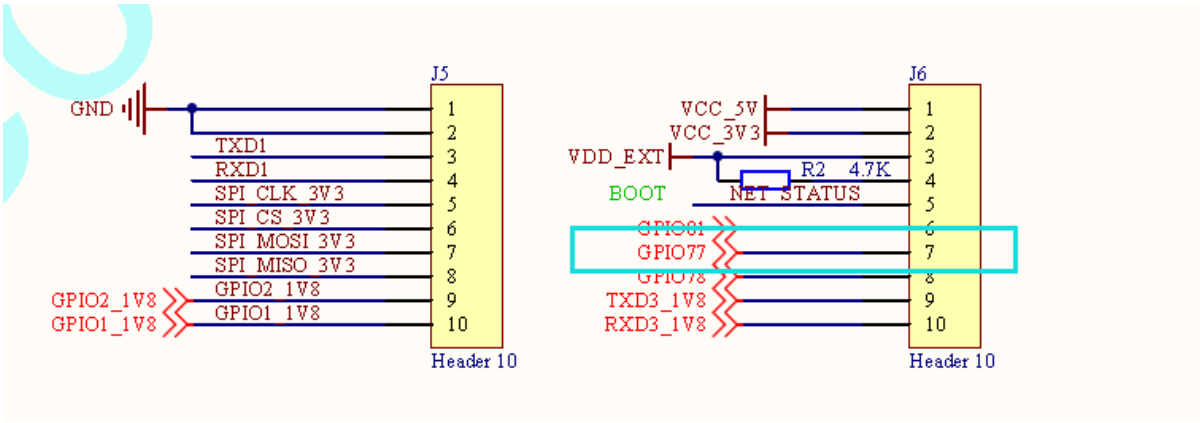


图 43：开发板引出的 GPIO7

步骤四：配置 QuecPython GPIO6，在开发板的 GPIO77 处查看配置是否生效。

6.6. usocket 使用

usocket 模块提供对 BSD 套接字接口的访问，支持对 socket 的地址绑定、监听、连接、数据接收和发送等通信方法。详细可参见：QuecPython API 类库文档 usocket 模块的使用。

- 代码示例

```
# 导入 usocket 模块
import usocket
import log

# 设置日志输出级别
log.basicConfig(level=log.INFO)
socket_log = log.getLogger("SOCKET")

# 创建一个 socket 实例
sock = usocket.socket(usocket.AF_INET, usocket.SOCK_STREAM)
# 解析域名
sockaddr = usocket.getaddrinfo('www.tongxinmao.com', 80)[0][-1]
# 建立连接
sock.connect(sockaddr)
# 向服务端发送消息
ret = sock.send('GET /News HTTP/1.1\r\nHost: www.tongxinmao.com\r\nAccept-Encoding:
deflate\r\nConnection: keep-alive\r\n\r\n')
socket_log.info('send %d bytes' % ret)
# 接收服务端消息
data = sock.recv(256)
```

```
socket_log.info('recv %s bytes:' % len(data))
socket_log.info(data.decode())

# 关闭连接
sock.close()
```

- 运行结果示例

6.7. QuecPython 是否支持队列

队列是一种先进先出的数据结构，主要操作包括入队、出队。入队的元素加入到对尾，从队头取出出队的元素。在 QuecPython 中可以使用 *list* 操作来模拟队列：

```
class queue:
def __init__(self):
        self.__alist = []

def push(self, value):
        self.__alist.insert(0, value)

def pop(self):
        return self.__alist.pop()

def size(self):
        return len(self.__alist)

def clean(self):
        self.__alist.clear()

def isEmpty(self):
```

```
        return self.__alist == []

    def showQueue(self):
        print(self.__alist)
```

运行：

```
if __name__ == '__main__':
q = queue()
    q.push(1)
    q.push("123")
    q.push("456")
    q.push(2)
    q.showQueue()
    print(q.pop())
    print(q.pop())
    print(q.pop())
    print(q.pop())
    q.showQueue()
```

6.8. socket 解析 IP 失败

- 问题现象

使用 socket、mqtt 等相关网络连接的 API 时会出现 IP 解析失败导致如下所示异常。

```
>>> import usocket
>>> socket = usocket.socket(usocket.AF_INET, usocket.SOCK_STREAM)
>>> sockaddr = usocket.getaddrinfo('www.tongxinmao.com',80)[0][-1]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
>>>
```

图 44：Socket 解析 IP 失败

- 问题定位

检查 SIM 卡是否注网成功以及检查该地址的有效性再次进行尝试。可通过 **AT+COPS?**查询 SIM 卡是否驻网成功。

6.9. 执行脚本文件提示语法错误

- **问题现象**

执行脚本文件出现如下如所示的语法错误提示

图 45：执行脚本文件提示语法错误

- **问题定位**

大概率是由于 Python 语法缩进不规范（4 个空格缩进），可检查代码格式缩进是否规范。另外，他推荐使用 PyCharm 或 VsCode 进行代码开发，IDE 会提示语法缩进与基本语法上的错误。

6.10. SIM 卡进行插拔后出现网络请求失败

目前 EC100Y-CN&EC600S-CN 暂不支持 SIM 卡热插拔功能，所以 SIM 卡在重新插拔后需要手动重启模块才能重新注网。

6.11. QuecPython 的源码文件是否安全

QPYcom 下载工具有代码混淆加密功能，确保用户程序不被直接暴露。

6.12. 在 QPYcom 操作没有任何反应

检查选择的串口是否正确并已打开。

6.13. 开发板送的流量卡是否可以混用

机卡绑定，第一次开机注网就绑定。

6.14. 编写 Python 代码用什么工具

推荐使用 pycharm 或 VScode，两款比较流行的 IDE，自带自动补全功能。

6.15. 板子接出的串口无法通信

要注意接口是 1.8V 还是 3.3V，电平匹配才能正常通信。

6.16. 接上 USB 线灯不亮

检查模块电压，保证模块的 3.8V 电压稳定，必要时用电池供电。

6.17. 小熊派右上方的 LED 点亮代码

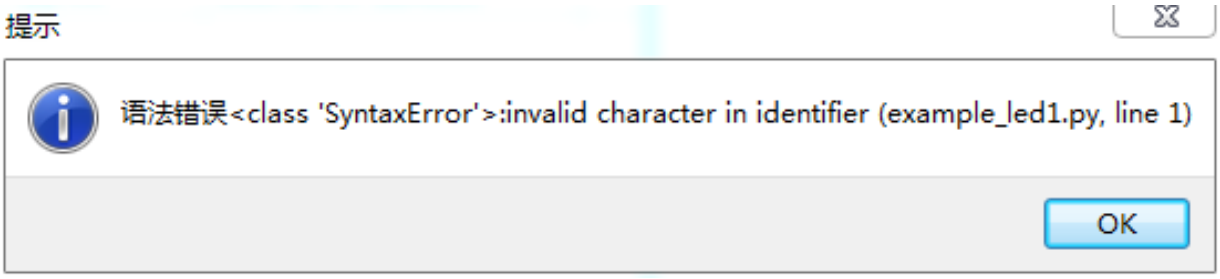
```
>>> from machine import Pin
>>> gpio = Pin(Pin.GPIO15, Pin.OUT, Pin.PULL_DISABLE, 0)
>>> gpio.write(1)
0
>>> gpio.read()
1
```

6.18. 是否可以删除 apn_cfg.json

apn_cfg.json 存放内置的 APN 列表，不可删除。

6.19. 拖动文件到模块出现语法错误

请检查“.py”文件的语法问题（一般为缩进不符合规范）。



6.20. 对 Socket 的[0][-1]的解释

```
>>> import usocket
>>> client = usocket.socket(usocket.AF_INET, usocket.SOCK_STREAM)
>>> sockaddr = usocket.getaddrinfo('www.tongxinmao.com',80)[0][-1]

>>> print(sockaddr)
('120.76.100.197', 80)
>>> sockaddr = usocket.getaddrinfo('www.tongxinmao.com',80)
>>> print(sockaddr)
[(2, 1, 0, 'www.tongxinmao.com', ('120.76.100.197', 80))]
>>>
```

举例说明：

```
a = ((A,B,C),(2),(3))
a[0] = (A,B,C)
a[0][-1] = C
```

6.21. 在 Win7 上运行 QPYcom 出现报错信息

- **故障现象**

Python 通过 pyinstaller 打包后，在个别 Win7 电脑运行失败，出现“failed to execute script pyi_rth_multiprocessing”提示信息。该问题仅在低版本 Win7 上运行会出现，在 Win10 上可正常移植程序。

- **故障分析**

可能是 Windows 某些 dll 文件版本过低，不支持高版本生成的 exe。

- **解决方法**

在 Win7 机器上单独打个 exe，然后在 Win7 运行并移植。或者升级 Win7 到 Win7 sp1 版本。

6.22. 对于 EC600SV1.1 的 QuecPython 板子，测试 TTS 功能注意事项

外设是喇叭，但是在 *tts* 模块中配置的是“话筒”。

由于 V1.1 上面加了一个功放，此时需要拉高模块（硬件拉高）socket 的 pin58（功放的使能脚），或者

通过在指令使能“audio_EN = Pin(Pin.GPIO11, Pin.OUT, Pin.PULL_PD, 1)”。

外接的喇叭是功率是有限制的，建议小于 8R 2W。

6.23. 对于多.py 文件的调用

- 调用 A 文件夹里面的 *B.py*。From A import B。

```
From A import B
```

- 调用 A 文件夹里面的 B 文件的 C 类。

方法 1：

```
From A.B import C
```

方法 2：

```
import sys
sys.path.append('A')
From B import C
```

7 附录 A 参考文档及术语缩写

表 2：参考文档

序号	文档名称	备注
[1]	《Quectel_QuecPython_QPYcom 工具使用说明》	QuecPython QPYcom 工具使用说明
[2]	《Quectel_EC100Y-CN_QuecOpen_硬件设计手册》	Quectel_EC100Y-CN_QuecOpen_ 硬件设计手册

表 3：术语缩写

缩写	英文全称	中文全称
API	Application Programming Interface	应用程序编程接口
APN	Access Point Name	接入点名称
DIAG	Diagnosis	诊断
DM	Device Manager	设备管理器
FOTA	Firmware Over-The-Air	空中下载软件升级
GPIO	General-Purpose Input/Output	通用型输入/输出
HTTP	Hypertext Transfer Protocol	超文本传输协议
IDE	Integrated Development Environment	集成开发环境
IP	Internet Protocol	网际互连协议
MQTT	Message Queuing Telemetry Transport	消息队列遥测传输
SIM	Subscriber Identity Module	用户身份识别模块
TTL	Transistor-Transistor Logic	晶体管-晶体管逻辑
TTS	Text To Speech	从文本到语音

TX	Transmit/Transmission	发送/传输
UART	Universal Asynchronous Receiver/Transmitter	通用异步收发传输器
USB	Universal Serial Bus	通用串行总线