RT-THREAD W60X SDK 开发手册

RT-THREAD 文档中心

上海睿赛德电子科技有限公司版权 ©2019

WWW.RT-THREAD.ORG

Thursday 21st February, 2019

目录

目录 **i**

1 前言 **1**

2 硬件篇 **2**

2.1 开发板整体介绍 . . . 2

2.1.1 W600 芯片介绍 . . . 2

2.1.2 W600_EV Board 介绍 . . . 3

2.2 开发板资源说明 . . . 4

2.2.1 复位按键 . . . 4

2.2.2 Arduino 标准接口 . . . 4

2.2.3 引出 IO 口 . . . 5

2.2.4 W600 芯片 . . . 5

2.2.5 WIFI 天线 . . . 5

2.2.6 RGB LED . . . 5

2.2.7 BOOT 按键 . . . 5

2.2.8 RGB LED 开关 . . . 5

2.2.9 UART0 . . . 5

2.2.10 UART1 . . . 5

2.2.11 POWER LED . . . 5

2.2.12 用户按键 . . . 6

3 软件篇 **7**

4 LED 闪烁例程 **8**

4.1 简介 . . . 8

4.2 硬件说明 . . . 8

4.3 软件说明 . . . 10
4.4 运行 . . . 10
4.4.1 编译 & 下载 . . . 10
4.4.2 运行效果 . . . 11
4.5 注意事项 . . . 11
4.6 引用参考 . . . 11

5 使用 MicroPython 控制硬件 12

5.1 简介 . . . 12
5.2 硬件说明 . . . 12
5.3 软件说明 . . . 12
5.4 运行 . . . 13
5.4.1 编译 & 下载 . . . 13
5.4.2 运行效果 . . . 14
5.5 MicroPython 基本功能 . . . 14
5.5.1 Python 语法与内建函数 . . . 14
5.5.1.1 使用 python 交互命令行 . . . 14
5.5.1.2 交互命令行的粘贴模式 . . . 15
5.5.1.3 MicroPython 内建模块 . . . 15
5.5.2 MicroPython 例程 . . . 16
5.6 注意事项 . . . 17
5.7 引用参考 . . . 17

6 中国移动 OneNET 云平台接入例程 18

6.1 简介 . . . 18
6.2 硬件说明 . . . 18
6.3 准备工作 . . . 18
6.3.1 创建设备 . . . 18
6.3.2 代码移植 . . . 19
6.3.2.1 保存设备信息 . . . 19
6.3.2.2 获取注册设备信息 . . . 20
6.3.2.3 获取设备信息 . . . 21
6.3.2.4 查询设备注册状态 . . . 21
6.4 软件说明 . . . 22

6.5 运行 24
6.5.1 编译 & 下载 24
6.5.2 连接无线网络 24
6.5.3 数据上传 25
6.5.4 命令控制 25
6.6 注意事项 27
6.7 引用参考 27

第 1 章

前言

开发手册包含两部分内容，包括实验平台硬件 W600_EV Board 的介绍和基于该平台的丰富示例。

第 2 章

硬件篇

本篇将详细介绍 W600_EV Board 硬件平台

2.1 开发板整体介绍

2.1.1 W600 芯片介绍

W600 芯片架构如下图所示：

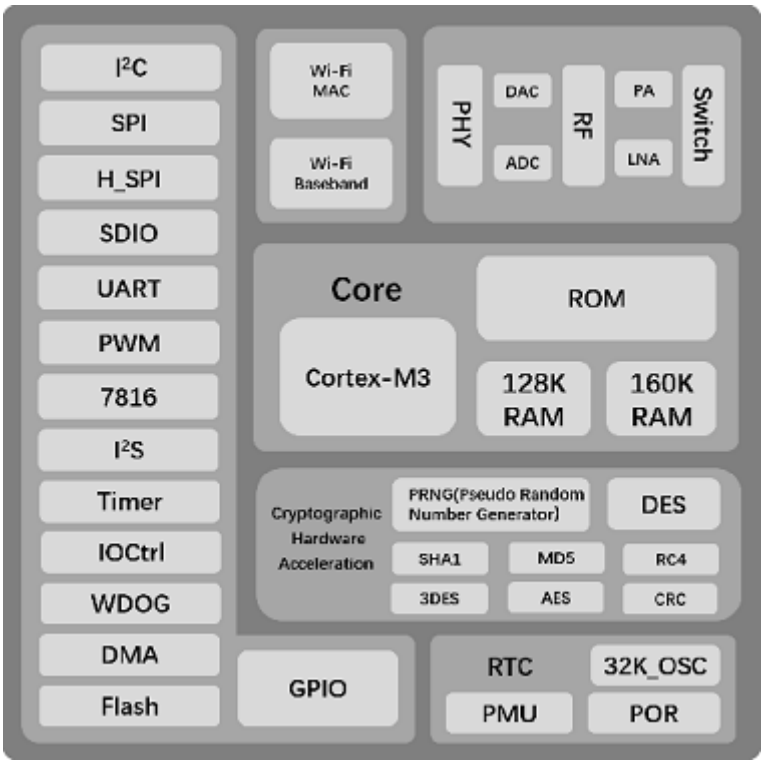

图 2.1: *W600 芯片架构*

W600 芯片资源如下：

- 芯片外观
  - QFN32 封装
- 芯片集成度
  - 集成 32 位嵌入式 Cortex-M3 处理器
  - 集成 288KByte 数据存储器
  - 集成 1MByte Flash
  - 集成 2.4G 射频收发器，满足 IEEE802.11 规范
  - 集成 PA/LNA/TR-Switch
  - 集成 32.768KHz 时钟振荡器
  - 集成电源管理电路
  - 集成通用加密硬件加速器，支持 PRNG(Pseudo random Number Generator)/ SHA1/ MD5/ RC4/ DES/ 3DES/ AES/ CRC 等多种加解密协议
  - 支持 3.3V 单电源供电
  - 集成 PS-Poll、U-APSD 低功耗管理
- 芯片接口
  - 集成 1 个 SDIO2.0 Device 控制器，支持 SDIO1 位/4 位/SPI 三种操作模式；工作时钟范围 0~50MHz
  - 集成 2 个 UART 接口，支持 RTS/CTS，波特率范围 38Kbps~2Mbps
  - 集成 1 个高速 SPI 设备控制器，工作时钟范围 0~50MHz
  - 集成 1 个 I²C 控制器，支持 100/400Kbps 速率
  - 集成 GPIO 控制器
  - 集成 PWM 控制器，支持 5 路 PWM 单独输出或者 2 路 PWM 输入
  - 集成双工 I²S 控制器，支持 32KHz 到 192KHz I²S 接口编解码
  - 集成 7816 接口，支持 ISO-7816-3 T=0/1 模式，支持 EVM2000 规范，并兼容串口功能
- 协议与功能
  - 支持 GB15629.11-2006、IEEE802.11 b/g/n/e/i/d/k/r/s/w
  - 支持 WAPI2.0
  - 支持 Wi-Fi WMM/WMM-PS/WPA/WPA2/WPS
  - 支持 Wi-Fi Direct
  - 支持 EDCA 信道接入方式
  - 支持 20/40M 带宽工作模式
  - 支持 AMPDU、AMSDU
  - 支持 IEEE802.11n MCS 0~7、MCS32 物理层传输速率档位，传输速率最高到 150Mbps
  - 支持 STA/AP/AP+STA 功能
  - 支持监听功能

2.1.2 W600_EV Board 介绍

W600_EV Board 主要资源如下图所示：

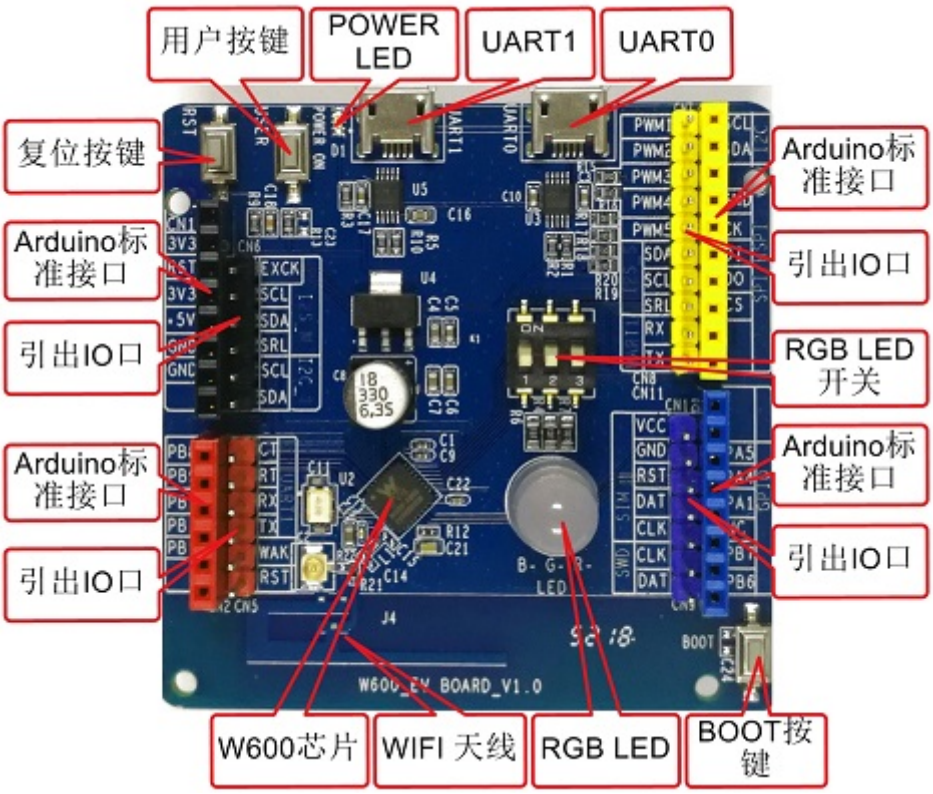

图 2.2: *W600_EV Board*

从图中可以看出，W600_EV Board 资源丰富，兼容标准 Arduino 接口，通过扩展 Arduino 外设板可进行大量的试验，非常适合物联网开发者与 WiFi 初学者使用。

开发板的特点包括：

- 小巧精致。开发板布局紧凑，丝印清晰美观，方便携带，可随时享受开发的乐趣。
- 接口丰富。开发板引出所有 IO，并设计有 arduino 接口，通过扩展 arduino 外设板，可进行大量实验，DIY 喜欢的产品。
- 简单易用。开发板仅一根 micro USB 线即可实现供电、下载。并且板载 WiFi 天线，方便接入网络。

2.2 开发板资源说明

对照板载资源图，按逆时针的顺序依次介绍。

2.2.1 复位按键

这是开发板的板载复位按键（RESET），用于复位开发板上的 W600 主芯片。

2.2.2 Arduino 标准接口

开发板上有一组标准的 arduino 接口（CN1、CN2、CN3、CN4）可方便的扩展 arduino 标准外设板。

2.2.3 引出 IO 口

开发板供引出 7 组 IO 口（CN6、CN5、CN7、CN8、CN11、CN12、CN9），供用户使用。

2.2.4 W600 芯片

芯片集成 Cortex-M3 内核，内置 1M Flash、288KB SRAM。集成射频收发前端 RF Transceiver，CMOS PA 功率放大器，基带处理器/媒体访问控制，支持 SDIO、SPI、UART、GPIO、I²C、PWM、I²S、7816 等接口, 支持多种加解密协议，如 PRNG/SHA1/MD5/RC4/DES/3DES/AES/CRC/RSA 等。支持多接口、多协议的无线局域网 IEEE802.11n（1T1R）。适用于智能家电、智能家居、无线音视频、智能玩具、医疗监护、工业控制等物联网应用领域。

2.2.5 WIFI 天线

这是开发板板载的一个 WIFI 天线（ANT）, 可以直接作为 WIFI 的天线使用。

2.2.6 RGB LED

这是开发板板载的一个 RGB 灯，通过 R（红）、G（绿）和 B（蓝色）三种颜色的组合我们可以实现各种不同的颜色。

2.2.7 BOOT 按键

这是开发板板载的一个 BOOT 按键，通过该按键可以下载.FLS 文件。

2.2.8 RGB LED 开关

这是开发板板载的一个 RGB LED 的拨码开关，将开关拨到ON可使得 RGB LED 和 IO 连通。

2.2.9 UART0

这是开发板板载的 UART0，该 UART 是默认的 log 输出口，可以下载固件（.FLS文件.img文件）, 并且也可以为开发板供电。

2.2.10 UART1

这是开发板板载的 UART1，该 UART 默认没有开启，仅支持（.img文件）的下载, 并且也可以为开发板供电。

2.2.11 POWER LED

这是开发板板载的一颗红色的 LED 灯（PWR），用于指示电源状态。在通电的时候，该灯会亮，否则不亮。通过这个 LED 可以判断开发板的上电情况。

2.2.12 用户按键

这是开发板板载的一个用户按键，供用户使用。

第 3 章

软件篇

在阅读过上面的硬件篇后，相信大家已经对开发板的硬件资源有了较为深入的了解，接下来我们将介绍 RT-Thread W60X SDK 的软件资源。

当前 RT-Thread W60X SDK 例程数量较少，但具有一定的代表性，能帮助开发者熟悉并使用上这套 SDK。后续将添加更多的例程供大家学习使用。

通过对这些例程的学习，你将了解到：

- 开发板硬件资源的使用方法
- 如何使用 RT-Thread 设备驱动框架
- 如何使用丰富的 IoT 软件包进行物联网应用开发

第 4 章

LED 闪烁例程

4.1 简介

本例程作为 SDK 的第一个例程，也是最简单的例程，类似于程序员接触的第一个程序 Hello World 一样简洁。

它的主要功能是让板载的 RGB-LED 中的蓝色 LED 不间断闪烁。

4.2 硬件说明

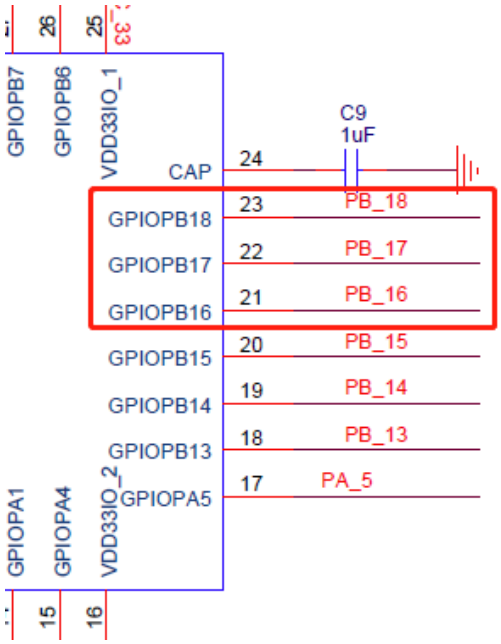

图 4.1: *LED 连接单片机引脚*

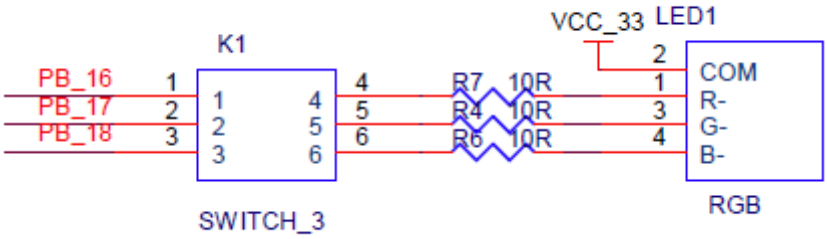

图 4.2: *LED 电路原理图*

如上图所示，RBG-LED 属于共阳 LED ，**阴极**通过拨码开关后，再连接到单片机的 21，22，23 号引脚上，其中蓝色 LED 对应 **23** 号引脚。拨动拨码开关，让单片机引脚与 RBG-LED 灯引脚相连。单片机引脚输出低电平即可点亮 LED ，输出高电平则会熄灭 LED。

LED 在开发板中的位置如下图所示：

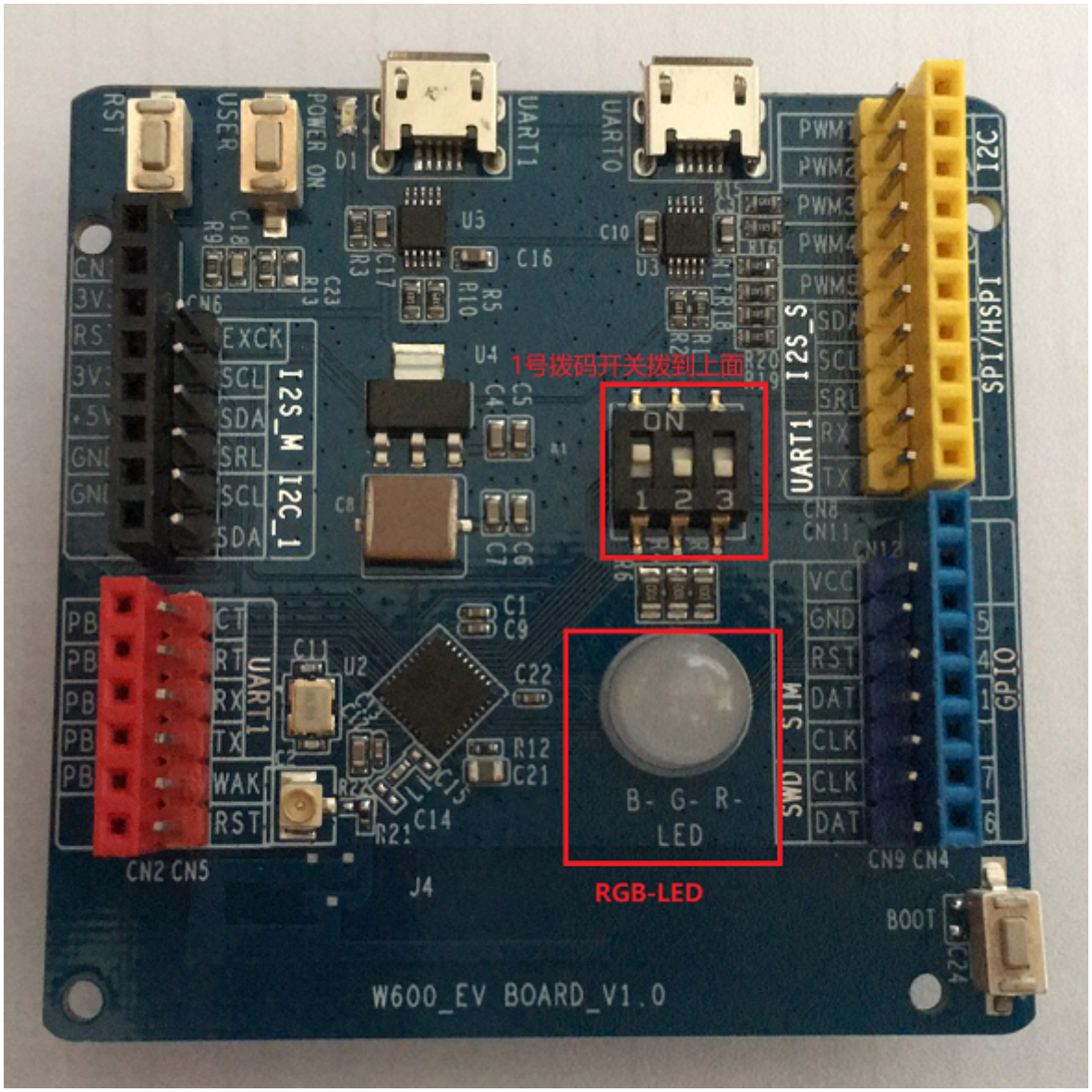

图 4.3: *LED 位置*

开发者需要手动将拨码开关的 1 号位置拨到上面，芯片的 23 号引脚才能跟 RGB-LED 的蓝色灯连接上。

4.3 软件说明

闪灯的源代码位于 `/examples/01_basic_led_blink/applications/main.c` 中。首先定义了一个宏 `LED_PIN`，与 LED 蓝色引脚 23 号相对应

```
/* using BLUE LED in RGB */
#define LED_PIN     (23)
```

在 main 函数中，将该引脚配置为输出模式，并在下面的 while 循环中，周期性（500 毫秒）开关 LED，同时输出一些日志信息。

```
int main(void)
{
    unsigned int count = 1;
    /* set LED pin mode to output */
    rt_pin_mode(LED_PIN, PIN_MODE_OUTPUT);

    while (count > 0)
    {
        /* led on */
        rt_pin_write(LED_PIN, PIN_LOW);
        rt_kprintf("led on, count: %d\n", count);
        rt_thread_mdelay(500);

        /* led off */
        rt_pin_write(LED_PIN, PIN_HIGH);
        rt_kprintf("led off\n");
        rt_thread_mdelay(500);

        count++;
    }

    return 0;
}
```

4.4 运行

4.4.1 编译 & 下载

- **MDK**：双击 `project.uvprojx` 打开 MDK5 工程，执行编译。
- **IAR**：双击 `project.eww` 打开 IAR 工程，执行编译。

编译完成后，将固件下载至开发板。

4.4.2 运行效果

按下复位按键重启开发板，观察开发板上 RBG-LED 的实际效果。正常运行后，蓝色 LED 会周期性闪烁，如下图所示：

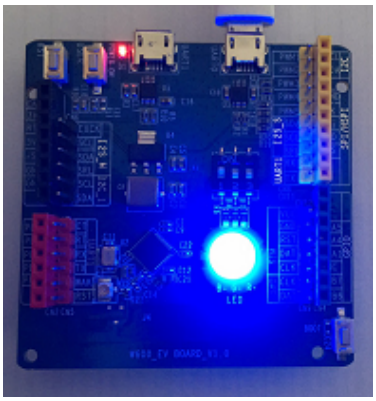

图 4.4: *RGB 蓝灯亮起*

此时也可以在 PC 端使用终端工具打开开发板的 `uart0` 串口，设置 115200 8 1 N 。开发板的运行日志信息即可实时输出出来。

```
led on, count: 1
led off
led on, count: 2
led off
led on, count: 3
led off
led on, count: 4
led off
led on, count: 5
led off
```

4.5 注意事项

开发板上的拨码开关分别连接 RGB-LED 和排针 CN7，如果 RGB-LED 灯不亮，请检测相应灯的拨码开关是否置于 ON 位置。

4.6 引用参考

- 《通用 GPIO 设备应用笔记》：docs/AN0002-RT-Thread-通用 GPIO 设备应用笔记.pdf
- 《RT-Thread 编程指南》：docs/RT-Thread 编程指南.pdf

第 5 章

使用 MicroPython 控制硬件

5.1 简介

MicroPython 是 Python 3 编程语言的一种精简而高效的实现，它包含 Python 标准库的一个子集，并被优化为在微控制器和受限环境中运行。它具有交互式提示、任意精度整数、闭包函数、列表解析、生成器、异常处理等高级特性，具有很强的可移植性。它能够帮助开发者快速控制各种硬件外设，不用再去研究底层硬件模块的使用方法，翻看寄存器手册。降低了开发难度，而且减少了重复开发工作，可以加快开发速度，提高开发效率。

本例程作将介绍如何在 rt-thread 上使用 MicroPython ，并通过命令行展示 MicroPython 代码的输入与运行，最后使用一个示例，演示使用 MicroPython 控制硬件。

5.2 硬件说明

本例程将点亮 LED 灯，因此请确保硬件平台上的 LED 灯能够正常工作。

5.3 软件说明

main 程序代码位于 `examples/02_micropython/applications/main.c` 中, 主要是为 MicroPython 的提供必要的运行环境。

在 main 函数中，主要完成以下几个任务：

- 创建 MTD 设备
- 挂载 LittleFS 文件系统
- 打开 MicroPython 命令交互界面

main 函数代码如下所示

```
int main(void)
{
```

```
    struct rt_device *flash_dev;

    /* 配置 wifi 工作模式 */
    rt_wlan_set_mode(RT_WLAN_DEVICE_STA_NAME, RT_WLAN_STATION);
    rt_wlan_set_mode(RT_WLAN_DEVICE_AP_NAME, RT_WLAN_AP);

    /* 初始化分区表 */
    fal_init();

    /* 在 fs_part 分区上创建一个 MTD 设备 */
    flash_dev = fal_mtd_nor_device_create(FS_PARTITION_NAME);
    if (flash_dev == NULL)
    {
        LOG_E("Can't create a mtd device on '%s' partition.\n", FS_PARTITION_NAME);
    }

    /* 挂载 LittleFS 文件系统 */
    if (dfs_mount(FS_PARTITION_NAME, "/", "lfs", 0, 0) == 0)
    {
        LOG_I("Filesystem initialized!\n");
    }
    else
    {
        /* 创建 LittleFS 文件系统 */
        dfs_mkfs("lfs", FS_PARTITION_NAME);
        /* 再次挂载 LittleFS 文件系统 */
        if (dfs_mount(FS_PARTITION_NAME, "/", "lfs", 0, 0) != 0)
        {
            LOG_E("Failed to initialize filesystem!");
        }
    }

    /* 打开 MicroPython 命令交互界面 */
    extern void mpy_main(const char *filename);
    mpy_main(NULL);

    LOG_D("MicroPython will reset by user");
    rt_hw_cpu_reset();
    return 0;
}
```

5.4 运行

5.4.1 编译 & 下载

- **MDK**：双击 `project.uvprojx` 打开 MDK5 工程，执行编译。
- **IAR**：双击 `project.eww` 打开 IAR 工程，执行编译。

编译完成后，将固件下载至开发板。

5.4.2 运行效果

在 PC 端使用终端工具打开开发板的 uart0 串口，设置 115200 8 1 N 。正常运行后，终端输出信息如下：

```
 \ | /
- RT -     Thread Operating System
 / | \     4.0.0 build Feb 15 2019
 2006 - 2018 Copyright by rt-thread team
lwIP-2.0.2 initialized!
[5] I/SAL_SOC: Socket Abstraction Layer initialize success.
[73] I/WLAN.dev: wlan init success
[108] I/WLAN.lwip: eth device init ok name:w0
[113] I/WLAN.dev: wlan init success
[148] I/WLAN.lwip: eth device init ok name:w1
[D/FAL] (fal_flash_init:63) Flash device | nor_flash | addr: 0x00000000 | len: 0
    x00100000 | blk_size: 0x00001000 |initialized finish.
[I/FAL] ==================== FAL partition table ====================
[I/FAL] | name      | flash_dev |   offset   |    length  |
[I/FAL] -------------------------------------------------------------
[I/FAL] | app       | nor_flash | 0x00010000 | 0x00080000 |
[I/FAL] | download  | nor_flash | 0x00090000 | 0x00060000 |
[I/FAL] | fs_part   | nor_flash | 0x000f0000 | 0x0000b000 |
[I/FAL] | easyflash | nor_flash | 0x000fb000 | 0x00001000 |
[I/FAL] =============================================================
[I/FAL] RT-Thread Flash Abstraction Layer (V0.3.0) initialize success.
[I/FAL] The FAL MTD NOR device (fs_part) created successfully
[235] I/main: Filesystem initialized!

MicroPython v1.10 on 2019-02-13; Universal python platform with RT-Thread
Type "help()" for more information.
>>>
```

此时 MicroPython 命令交互界面就已经启动，可以通过命令行与 MicroPython 进行交互。下面将使用一个示例展示如何使用 MicroPython 控制硬件。

5.5 MicroPython 基本功能

5.5.1 Python 语法与内建函数

5.5.1.1 使用 python 交互命令行

- MicroPython 是 Python 3 编程语言的一种精简而高效的实现，语法和 Python 3 相同，并带有丰富的内建函数，使用 MicroPython 交互命令行即可运行 Python 代码。

在交互命令行输入 `print('hello RT-Thread!')`，然后输入回车，将运行这行语句，在下一行输出 hello RT-Thread! 字样。运行效果如下：

图 5.1: *elect_micropytho*

5.5.1.2 交互命令行的粘贴模式

`MicroPython` 比一般的 python 交互环境多了一个特别的**粘贴模式**，可以一次粘贴输入多行 python 代码。

- 按下 `Ctrl-E` 组合键，进入粘贴模式，界面上会出现提示：`paste mode; Ctrl-C to cancel, Ctrl-D to finish` 。该模式下可一次粘贴多行代码。
- 按下 `Ctlr-D` 组合键，退出粘贴模式。同时粘贴输入的代码也会自动执行。
- 按下 `Ctrl-C` 组合键，终止正在执行的程序。

注意：进入粘贴模式后，不要使用 `Ctrl-C` 粘贴代码。可使用鼠标右键进行粘贴。

使用粘贴模式执行下面代码：

```
for i in range(1,10):
    print(i)
```

执行效果如下：

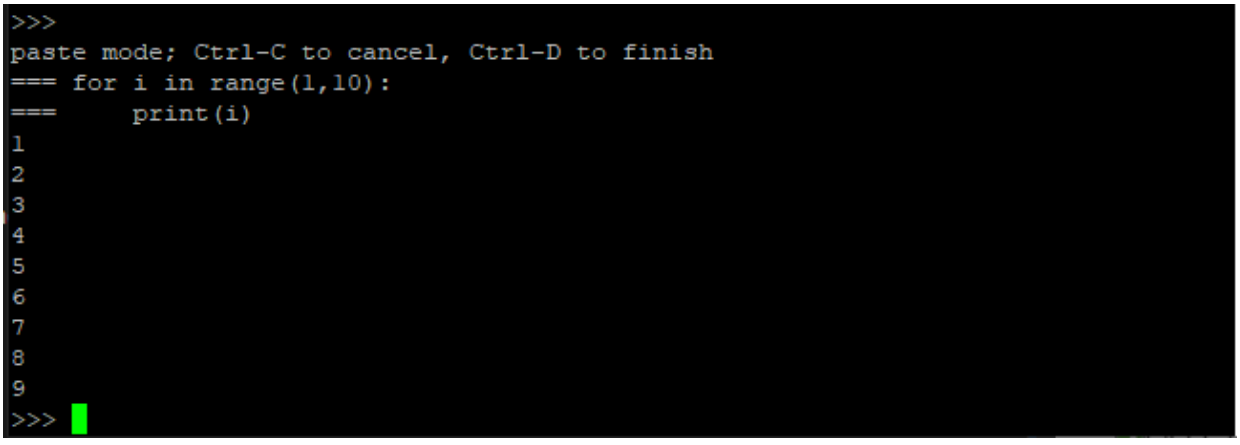

图 5.2: *elect_micropytho*

5.5.1.3 MicroPython 内建模块

MicroPython 提供丰富的内建模块用来完成相关的程序功能。同时 RT-Thread 也提供了 `rtthread` 模块用来返回系统运行相关的信息。

- 使用 rtthread 模块查看当前运行线程，调用方法及效果如下图所示：

```
>>> import rtthread
>>> rtthread.stacks_analyze()
thread   pri  status      sp     stack size max used left tick  error
-------- ---  ------- ---------- ---------- ------  ---------- ---
task-05   14  suspend 0x000000b4 0x00000200    35%   0x00000014 000
task-04   13  suspend 0x0000008c 0x00002000    01%   0x00000014 000
task-03   10  suspend 0x0000008c 0x00000200    27%   0x00000014 000
task-02    7  suspend 0x000000b4 0x00000640    11%   0x00000014 000
task-01    4  suspend 0x0000009c 0x000004b0    13%   0x00000014 000
task-00    5  suspend 0x000000ac 0x00000320    21%   0x00000014 000
tshell    20  suspend 0x000000ec 0x00001000    05%   0x00000009 000
wlan_job  22  suspend 0x00000068 0x00000800    05%   0x0000000a 000
tcpip     10  suspend 0x000000b4 0x00000400    37%   0x00000014 000
etx       12  suspend 0x00000094 0x00000400    20%   0x00000010 000
erx       12  suspend 0x00000094 0x00000400    14%   0x00000010 000
tidle0    31  ready   0x0000005c 0x00000100    35%   0x0000000b 000
timer      4  suspend 0x00000074 0x00000200    22%   0x00000009 000
main      10  running 0x00000304 0x00002000    14%   0x00000014 000
>>>
```

图 5.3: *elect_micropytho*

- 使用 time 模块进行毫秒延时，调用方法及效果如下图所示：

```
>>>
>>> import time
>>> time.sleep_ms(2000)
>>>
```

图 5.4: *elect_micropytho*

5.5.2 MicroPython 例程

通过 MicroPython 可以用非常简单的方式来控制开发板的硬件资源，使用一个简单的例子来说明：

- W600 开发板中: 第 23 号 pin 为蓝色 LED 灯。下面代码将周期闪烁 LED 灯。

```
import time
from machine import Pin

LED = Pin(("LED1", 23), Pin.OUT_PP)      #将第 23 号 Pin 设备设置为输出模式
while True:
    LED.value(1)
    time.sleep_ms(500)
    LED.value(0)
    time.sleep_ms(500)
```

针对自己的开发板修改引脚号，将以上脚本使用**粘贴模式**输入，即可看到 LED 灯按照指定的频率闪烁。使用 Ctrl-C 可以取消当前正在运行程序。

5.6 注意事项

- 想要了解更多的 MicroPython 软件包的功能，可以查阅 MicroPython 用户手册。

5.7 引用参考

《RT-Thread 编程指南》：docs/RT-Thread 编程指南.pdf

《MicroPython 用户手册》：docs/UM1011-RT-Thread-MicroPython 用户手册

第 6 章

中国移动 OneNET 云平台接入例程

本例程演示如何使用 RT-Thread 提供的 onenet 软件包接入中国移动物联网开放平台，介绍如何通过 MQTT 协议接入 OneNET 平台，初次使用 OneNET 平台的用户请先阅读《OneNET 用户手册》。

6.1 简介

OneNET 平台是中国移动基于物联网产业打造的生态平台，具有高并发可用、多协议接入、丰富 API 支持、数据安全存储、快速应用孵化等特点。OneNET 平台可以适配各种网络环境和协议类型，现在支持的协议有 LWM2M（NB-IOT）、EDP、MQTT、HTTP、MODBUS、JT808、TCP 透传、RGMP 等。用户可以根据不同的应用场景选择不同的接入协议。

onenet 软件包是 RT-Thread 针对 OneNET 平台做的适配，通过这个软件包可以让设备在 RT-Thread 上非常方便的连接 OneNet 平台，完成数据的发送、接收、设备的注册和控制等功能。

6.2 硬件说明

6.3 准备工作

6.3.1 创建设备

在使用本例程前需要在 OneNET 平台注册账号，并在帐号里创建产品，具体的流程参考《OneNET 示例说明》。产品创建完成后，记录下产品概况页面的产品 ID 和 APIKey。

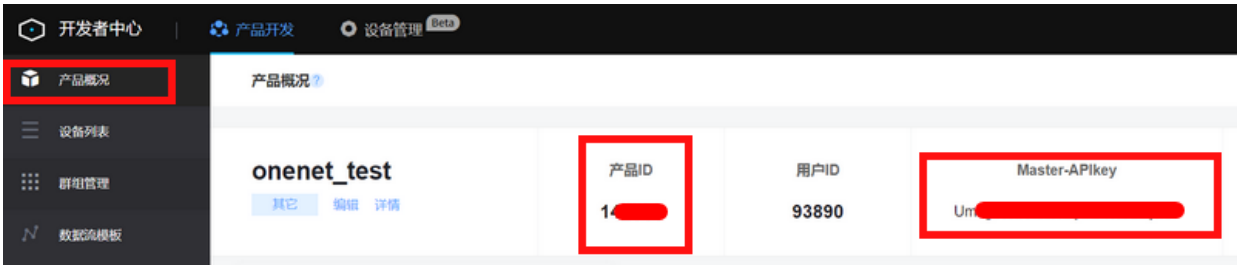

图 6.1: 设备信息

切换到设备管理界面，记录下设备注册码。

图 6.2: *环境注册码*

打开 /examples/03_iot_cloud_onenet/rtconfig.h,找到 ONENET_REGISTRATION_CODE,ONENET_INFO_PROID，ONENET_MASTER_APIKEY 这三个宏定义，将原来的内容替换成刚刚记录下来的环境注册码，产品 ID 和 APIKey，然后保存文件。

6.3.2 代码移植

设备注册，设备上线，需要用到一些需要移植的接口函数，下面将具体介绍需要用到的四个接口函数。IoT Board 关于 OneNET 的移植代码位于 /examples/03_iot_cloud_onenet/applications/main.c 文件中。

6.3.2.1 保存设备信息

注册设备成功后，OneNET 平台会返回设备 ID 和 api key，用户需要将这两个信息保存起来，以便在下次开机时能读取这两个信息用于登录 OneNET 平台。除了保存 2 个设备信息外，用户还应该保存个已经注册成功的标志位，用来在开机时判断设备是否已经注册。

```
rt_err_t onenet_port_save_device_info(char *dev_id, char *api_key)
{
    EfErrCode err=EF_NO_ERR;

    /* 保存设备 ID */
    err = ef_set_and_save_env("dev_id",dev_id);
    if(err != EF_NO_ERR)
    {
        rt_kprintf("save device info(dev_id : %s) failed!\n", dev_id);
        return -RT_ERROR;
    }

    /* 保存设备 api_key */
    err = ef_set_and_save_env("api_key", api_key);
```

```
    if(err != EF_NO_ERR)
    {
        rt_kprintf("save device info(api_key : %s) failed!\n", api_key);
        return -RT_ERROR;
    }

    /* 保存环境变量：已经注册 */
    err = ef_set_and_save_env("already_register","1");
    if(err != EF_NO_ERR)
    {
        rt_kprintf("save already_register failed!\n");
        return -RT_ERROR;
    }

    return RT_EOK;
}
```

6.3.2.2 获取注册设备信息

设备注册需要提供设备名字和鉴权信息，这里取设备的的 mac 地址用作设备名字和鉴权信息，除了将 mac 地址赋值给 2 个入参外，还应该作为鉴权信息保存起来，用于下次开机登录 OneNET 平台。

```
rt_err_t onenet_port_get_register_info(char *dev_name, char *auth_info)
{
    rt_uint32_t mac_addr[2] = {0};
    EfErrCode err = EF_NO_ERR;

    /* get mac addr */
    if (rt_wlan_get_mac((rt_uint8_t *)mac_addr) != RT_EOK)
    {
        rt_kprintf("get mac addr err!! exit\n");
        return -RT_ERROR;
    }

    /* set device name and auth_info */
    rt_snprintf(dev_name, ONENET_INFO_AUTH_LEN, "%d%d", mac_addr[0], mac_addr[1]);
    rt_snprintf(auth_info, ONENET_INFO_AUTH_LEN, "%d%d", mac_addr[0], mac_addr[1]);

    /* save device auth_info */
    err = ef_set_and_save_env("auth_info", auth_info);
    if (err != EF_NO_ERR)
    {
        rt_kprintf("save auth_info failed!\n");
        return -RT_ERROR;
    }

    return RT_EOK;
}
```

6.3.2.3 获取设备信息

获取用于登录的设备信息功能的代码如下所示：

```
rt_err_t onenet_port_get_device_info(char *dev_id, char *api_key, char *auth_info)
{
    char *info =RT_NULL;

    /* 获取设备 ID */
    info = ef_get_env("dev_id");
    if( info == RT_NULL)
    {
        rt_kprintf("read dev_id failed!\n");
        return -RT_ERROR;
    }
    else
    {
        rt_snprintf(dev_id,ONENET_INFO_AUTH_LEN,"%s",info);
    }

    /* 获取 api_key */
    info = ef_get_env("api_key");
    if( info == RT_NULL)
    {
        rt_kprintf("read api_key failed!\n");
        return -RT_ERROR;
    }else
    {
        rt_snprintf(api_key,ONENET_INFO_AUTH_LEN,"%s",info);
    }

    /* 获取设备鉴权信息 */
    info = ef_get_env("auth_info");
    if( info == RT_NULL)
    {
        rt_kprintf("read auth_info failed!\n");
        return -RT_ERROR;
    }else
    {
        rt_snprintf(auth_info,ONENET_INFO_AUTH_LEN,"%s",info);
    }

    return RT_EOK;
}
```

6.3.2.4 查询设备注册状态

检查设备是否已经注册的代码如下所示：

```
rt_bool_t onenet_port_is_registed(void)
{
    char *already_register = RT_NULL;

    /* 检查设备是否已经注册 */
    already_register = ef_get_env("already_register");
    if (already_register == RT_NULL)
    {
        return RT_FALSE;
    }

    return already_register[0] == '1' ? RT_TRUE : RT_FALSE;
}
```

6.4 软件说明

本例程主要实现了每隔 5s 往 OneNET 平台上传一次环境光强度，并可以执行 OneNET 下发命令的功能，程序代码位于 /examples/03_iot_cloud_onenet/applications/main.c 文件中。

在 main 函数中，主要完成了以下几个任务：

- 初始化 LED 管脚
- 注册 OneNET 启动函数为 WiFi 连接成功的回调函数

当 WiFi 连接成功后，会调用 /examples/03_iot_cloud_onenet/packages/onenet-1.0.0/src/onenet_mqtt.c 文件中的 onenet_mqtt_init() 函数，该函数会获取设备信息完成设备上线的任务。如果设备未注册，会自动调用注册函数完成注册。

OneNET 设备上线后，会自动执行 /examples/03_iot_cloud_onenet/applications/main.c中的 onenet_upload_cycle() 函数，函数会设置命令响应的回调函数，并启动一个 onenet_send 的线程，线程会每隔 5 秒随机得到一个环境光强度数据，并将这个数据上传到 OneNET 的 light 数据流中，发送 100 次后线程自动结束。上传数据的代码如下所示：

```
static void onenet_upload_entry(void *parameter)
{
    int value = 0;
    int i = 0;

    srand((int)rt_tick_get());
    /* upload ambient light value to topic light*/
    for (i = 0; i < NUMBER_OF_UPLOADS; i++)
    {
        value = rand() % 100;

        if (onenet_mqtt_upload_digit("light", value) < 0)
        {
            rt_kprintf("upload has an error, stop uploading\n");
```

```
            break;
        }
        else
        {
            rt_kprintf("buffer : {\"light\":%d}\n", value);
        }

        rt_thread_mdelay(5 * 1000);
    }
}
```

命令响应回调函数里主要完成的是命令打印，命令匹配的任务。当匹配到 ledon 和 ledoff 这两个命令时，会执行打开 led 和关闭 led 的动作，并向云端发送响应。具体代码如下所示：

```
static void onenet_cmd_rsp_cb(uint8_t *recv_data, size_t recv_size, uint8_t **
    resp_data, size_t *resp_size)
{
    char res_buf[20] = { 0 };

    rt_kprintf("recv data is %.s\n", recv_size, recv_data);

    /* 命令匹配 */
    if (rt_strncmp(recv_data, "ledon", 5) == 0)
    {
        /* led on */
        rt_pin_write(LED_PIN, PIN_LOW);

        rt_snprintf(res_buf, sizeof(res_buf), "led is on");

        rt_kprintf("led is on\n");
    }
    else if (rt_strncmp(recv_data, "ledoff", 6) == 0)
    {
        /* led off */
        rt_pin_write(LED_PIN, PIN_HIGH);

        rt_snprintf(res_buf, sizeof(res_buf), "led is off");

        rt_kprintf("led is off\n");
    }

    /* 开发者必须使用 ONENET_MALLOC 为响应数据申请内存 */
    *resp_data = (uint8_t *) ONENET_MALLOC(strlen(res_buf) + 1);

    strncpy(*resp_data, res_buf, strlen(res_buf));

    *resp_size = strlen(res_buf);
}
```

6.5 运行

6.5.1 编译 & 下载

- **MDK**：双击 `project.uvprojx` 打开 MDK5 工程，执行编译。
- **IAR**：双击 `project.eww` 打开 IAR 工程，执行编译。

编译完成后，将开发板的 ST-Link USB 口与 PC 机连接，然后将固件下载至开发板。

程序运行日志如下所示：

```
  \ | /
- RT -     Thread Operating System
 / | \     4.0.0 build Feb 13 2019
 2006 - 2018 Copyright by rt-thread team
lwIP-2.0.2 initialized!
[5] I/SAL_SOC: Socket Abstraction Layer initialize success.
[75] I/WLAN.dev: wlan init success
[113] I/WLAN.lwip: eth device init ok name:w0
[118] I/WLAN.dev: wlan init success
[154] I/WLAN.lwip: eth device init ok name:w1
[D/FAL] (fal_flash_init:63) Flash device | nor_flash | addr: 0x00000000 | len: 0
    x00100000 | blk_size: 0x00001000 |initialized finish.
[I/FAL] ==================== FAL partition table ====================
[I/FAL] | name      | flash_dev |   offset   |    length   |
[I/FAL] -------------------------------------------------------------
[I/FAL] | app       | nor_flash | 0x00010000 | 0x00080000 |
[I/FAL] | download  | nor_flash | 0x00090000 | 0x00060000 |
[I/FAL] | fs_part   | nor_flash | 0x000f0000 | 0x0000b000 |
[I/FAL] | easyflash | nor_flash | 0x000fb000 | 0x00001000 |
[I/FAL] =============================================================
[I/FAL] RT-Thread Flash Abstraction Layer (V0.3.0) initialize success.
[Flash] EasyFlash V3.3.0 is initialize success.
[Flash] You can get the latest version on https://github.com/armink/EasyFlash .
msh />
```

6.5.2 连接无线网络

程序运行后会进行 MSH 命令行，等待用户配置设备接入网络。使用 MSH 命令 `wifi join ssid password` 配置网络，如下所示：

```
msh />wifi join ssid password
join ssid:ssid
[I/WLAN.mgnt] wifi connect success ssid:ssid_test
.......
msh />[I/WLAN.lwip] Got IP address : 152.10.200.224
```

6.5.3 数据上传

在 WiFi 连接成功后，开发板会自动连接 OneNET 平台，如果是第一次连接云平台，会自动注册设备，如下所示：

```
msh />[D/ONENET] (response_register_handlers:266) response is {"errno":0,"data":{"
    device_id":"42940432","key":"GZz=nP9xGMENrsBFFPMDUUe66Q8="},"error":"succ"}    #
    注册返回的设备信息
[D/ONENET] (mqtt_connect_callback:87) Enter mqtt_connect_callback!
[D/MQTT] ipv4 address port: 6002
[D/MQTT] HOST = '183.230.40.39'
[I/ONENET] RT-Thread OneNET package(V1.0.0) initialize success.
[I/WLAN.lwip] Got IP address : 152.10.200.224
[I/MQTT] MQTT server connect success
[D/ONENET] (mqtt_online_callback:92) Enter mqtt_online_callback!
[D/ONENET] (onenet_upload_entry:82) buffer : {"light":20}    #开始上传数据
[D/ONENET] (onenet_upload_entry:82) buffer : {"light":28}
```

打开 OneNET 平台，在**设备列表**页面，选择**数据流展示**，点击展开 **light** 数据流，可以看到刚刚上传的数据信息。

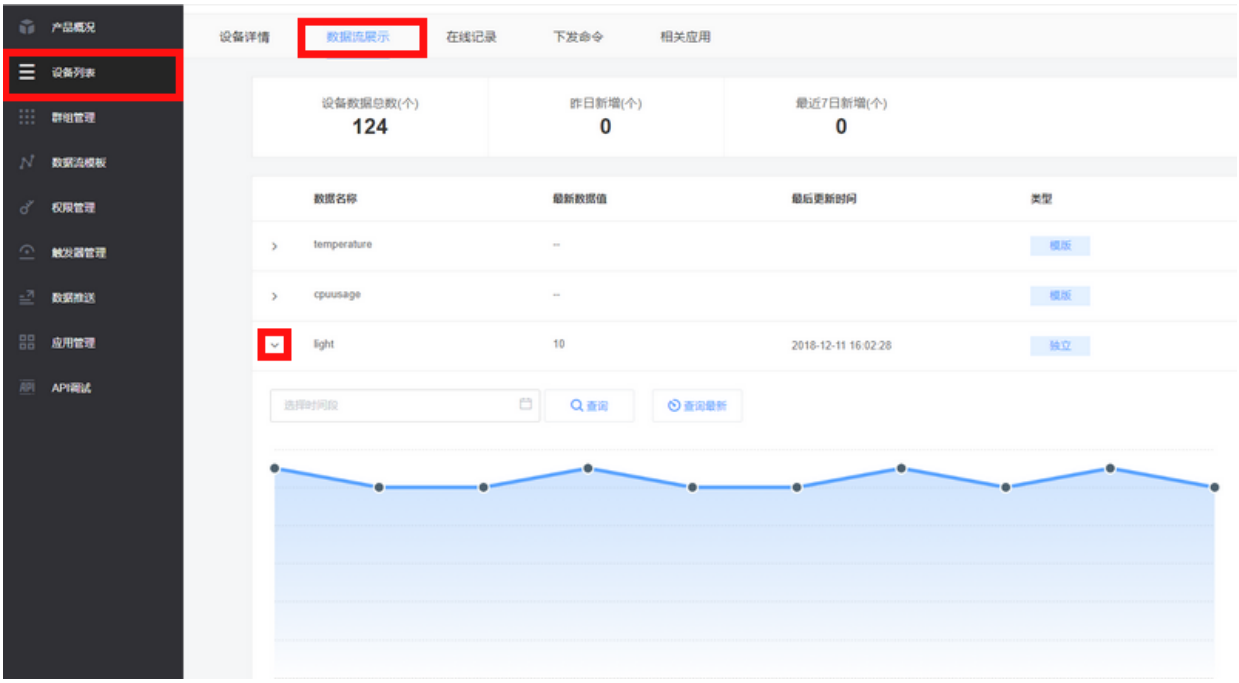

图 6.3: *数据流*

6.5.4 命令控制

在**设备列表**界面，选择**下发命令**，点击蓝色的**下发命令**按钮，会弹出一个命令窗口，我们可以下发命令给开发板。例程支持 ledon 和 ledoff 两个命令，板子收到这两个命令后会打开和关闭开发板上的红色 led。示例效果如下所示：

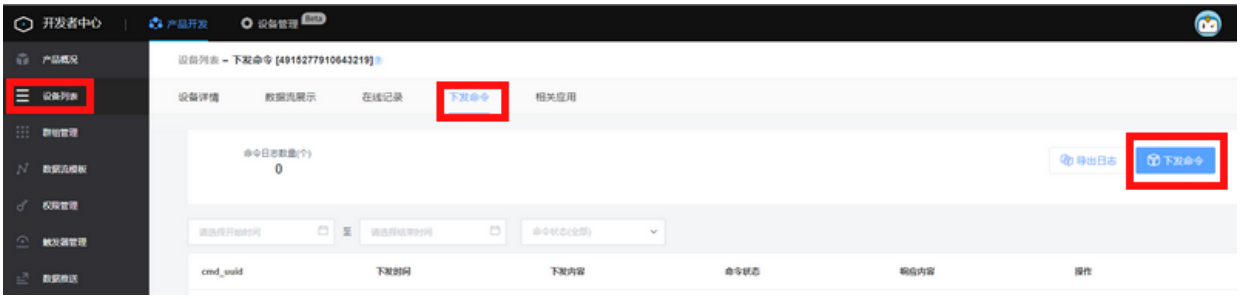

图 6.4: 设备控制界面

图 6.5: 发送命令

```
[D/ONENET] (mqtt_callback:62) topic $creq/1798e855-c334-5f03-a81a-5d8f587a0bc9
    receive a message
[D/ONENET] (mqtt_callback:64) message length is 5
[D/ONENET] (onenet_cmd_rsp_cb:212) recv data is ledon
[D/ONENET] (onenet_cmd_rsp_cb:248) led is on
```

6.6 注意事项

- 使用本例程前请先阅读《OneNET 用户手册》。
- 使用本例程前请先修改 `rtconfig.h` 里的三个 OneNET 平台宏定义 (`ONENET_REGISTRATION_CODE` ,`ONENET_INFO_PROID`,`ONENET_MASTER_APIKEY`)。

6.7 引用参考

- 《GPIO 设备应用笔记》：docs/AN0002-RT-Thread-通用 GPIO 设备应用笔记.pdf
- 《OneNET 用户手册》：docs/UM1003-RT-Thread-OneNET 用户手册.pdf
- 《RT-Thread 编程指南》：docs/RT-Thread 编程指南.pdf