UPS Plus 中文说明书

购买链接

https://52pi.com/products/52pi-ups-board-with-rtc-coulometer-for-raspberry-pi

描述

UPS Plus 是新一代的 UPS 电源管理模块，支持 OTA 固件升级和内置 RTC（实时时钟）。

并且板上有两个 INA219 芯片用于检测电池电压和 UPS Plus 到 Raspberry Pi 的输出电压。

警告：在使用 UPS 期间，不要将电源供应器插入 Raspberry Pi 的 USB-C 端口，可能会损坏 UPS!

兼容性

Raspberry 型号	兼容性
Raspberry Pi A	是
Raspberry Pi A+	是
Raspberry Pi B	是
Raspberry Pi B+	是
Raspberry Pi 2B	是
Raspberry Pi 3B	是
Raspberry Pi 3B+	是
Raspberry Pi 4B	是
Raspberry Pi Zero	否
Raspberry Pi Zero W	否

特点

- 易于安装
- 增强的电源管理
- 远程 OTA 固件升级
- 可编程 BACK-TO-AC 自动电源启动
- 可编程采样周期
- I2C 通信
- 独立 RTC
- 支持堆叠电池设计

规格

- 监控 Raspberry Pi 电源端口的电流/电压。
- 电池端口电流/电压监控，支持充放电双向监控。
- 独立 RTC 功能。
- OTA 功能（支持强制升级模式和主动升级模式）。
- 功率估算
    - **注意**：至少需要完成一个完整的充放电周期！
- 可调采样周期。
- 支持 FCP、AFC、SFCP 快充协议。
- 支持 BC1.2 充电协议。
- 电池温度监控
    - **注意**：强制温度保护无法关闭，阈值：65 度！
- 可编程电源电压检测器 (PVD) 功能，默认值：3.6V。
- 来电自动启动功能。
- 断电记忆功能。
- 可编程关机 / 强制重启。
- 运行时间统计。
- 抗静电保护。
- 线性补偿范围放电容量：5V 4.5A。
- 非线性补偿范围放电容量：5V 8A。
- 充电容量：4.5V 5A/5V 2.5A/9V 2A/12V 1.5A。
- 堆叠电池设计
    - **注意**：堆叠电源板需要特殊配件，不支持非官方配件。
- 使用 I2C 通信，不占用额外端口。
- 支持 4.2V、4.35V、4.4V、4.5V 18650 锂电池
    - **注意**：不同类型的电池不能混用！
- 售后数据遥测。

注册映射

USB Plus V5.0 注册映射图

- 0x17 - 操作模式
  - RO - 只读，RW - 读写

USB Plus V5.0 注册映射图

地址范围	功能	范围	单位	读写属性
0x01 - 0x02	UPS的MCU电压	2400 - 3600	mV	只读 (RO)
0x03 - 0x04	Pogopin底部电压	0 - 5500	mV	只读 (RO)
0x05 - 0x06	电池端口电压	0 - 4500	mV	只读 (RO)
0x07 - 0x08	USB-C充电端口电压	0 - 13500	mV	只读 (RO)
0x09 - 0x0A	MicroUSB充电端口电压	0 - 13500	mV	只读 (RO)
0x0B - 0x0C	电池温度	-20 - 65	℃	只读 (RO)
0x0D - 0x0E	满电电压	0 - 4500	mV	读写 (RW)
0x0F - 0x10	空电电压	0 - 4500	mV	读写 (RW)
0x11 - 0x12	保护电压	0 - 4500	mV	读写 (RW)
0x13 - 0x14	电池剩余	0 - 100	%	只读 (RO)
0x15 - 0x16	采样周期	1 - 1440	分钟	读写 (RW)
0x17	电源状态/操作模式	0/1	布尔	只读 (RO)
0x18	关机倒计时	0(假)/10 - 255	秒	读写 (RW)
0x19	BACK-TO-AC自动电源启动	0/1	布尔	读写 (RW)
0x1A	重启倒计时	0(假)/10 - 255	秒	读写 (RW)
0x1B	恢复出厂默认设置	0/1	布尔	读写 (RW)
0x1C - 0x1F	累积运行时间	0 - 2147483647	秒	只读 (RO)
0x20 - 0x23	累积充电时间	0 - 2147483647	秒	只读 (RO)
0x24 - 0x27	运行时间	0 - 2147483647	秒	只读 (RO)
0x28 - 0x29	版本	1	固定	只读 (RO)
0x2A	(FW.V5+) 用户自编程电池参数	0/1	布尔	读写 (RW)
0x2B - 0xEF	[保留]	NA	NA	读写 (RW)

0xF0 - 0xFB	序列号	设备 UID	固定	只读 (RO)
0xFC - 0xFF	工厂测试	NA	NA	读写 (RW)

0x18 - OTA固件升级模式

地址范围	功能
0x01 - 0x10	加密缓冲区
0x11 - 0xEF	[保留]
0xF0 - 0xFB	序列号
0xFC - 0xFF	工厂测试

请注意，表格中 "RO" 代表只读（Read Only），"RW" 代表可读写（Read & Write），"NA" 代表不适用或未指定，"Bool" 代表布尔值，"Fixed" 代表固定值。

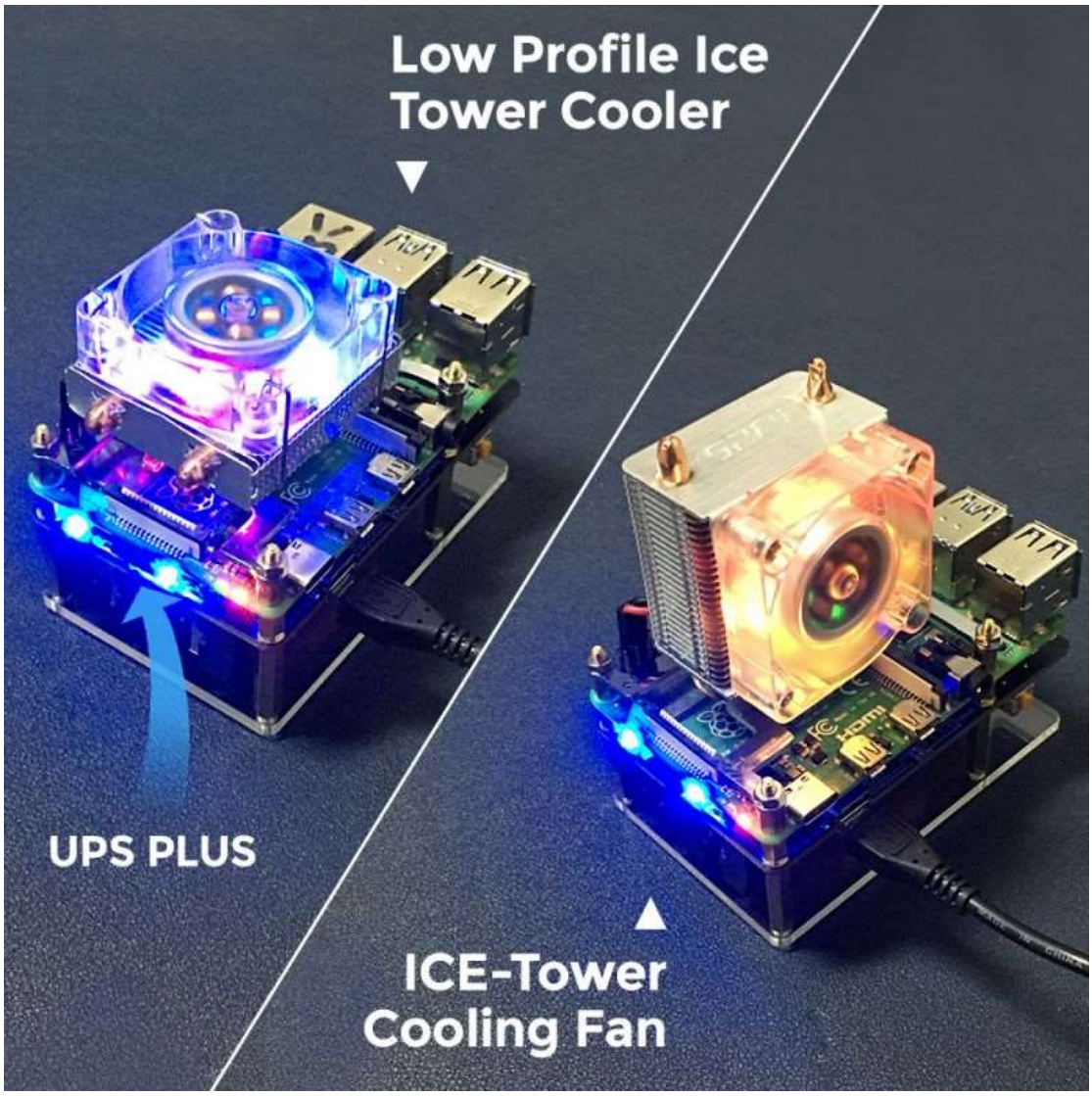
画廊

- 产品外观


USB-C
PORT


USB
Micro Port

- 应用场景



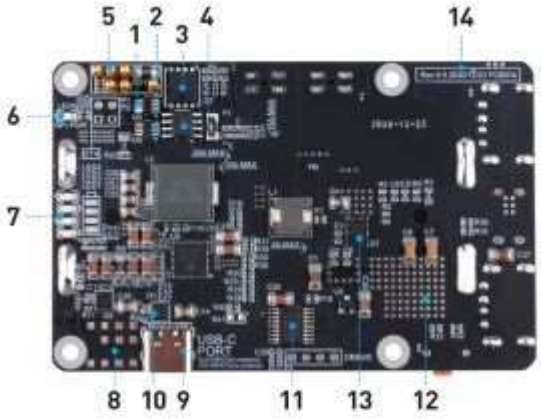
PCB 绘图

PCBDOC 文件:

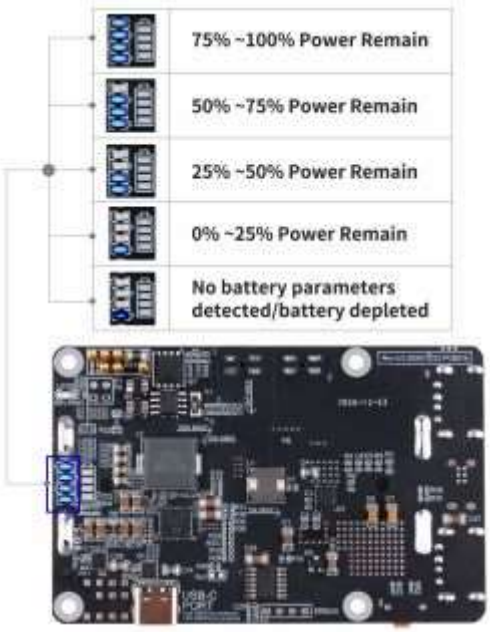
https://wiki.52pi.com/index.php?title=File:UPS_PLUS_PCBDOC.zip

功能区域显示

1. Stack function
2. USB A port output (5V 2A total)
3. USB C port output (5V 4A total/shared with USB-A)
4. Battery interface
5. Positioning mounting hole
6. USB-Micro fast charging interface
7. Multi-function buttons



1. Raspberry Pi voltage/current/power sampling
2. Battery voltage/current/power sampling
3. High performance MOS switch
4. RTC
5. Raspberry Pi power supply interface
6. Raspberry Pi power supply LED indicator
7. Battery LED indicator
8. Copper Cooling
9. USB-C fast charging interface
10. Battery management unit
11. UPS 32-bit processor
12. Multi-area Cooling
13. 5V power supply management unit
14. Hardware version number mark



LED 灯状态定义

LED 灯状态定义

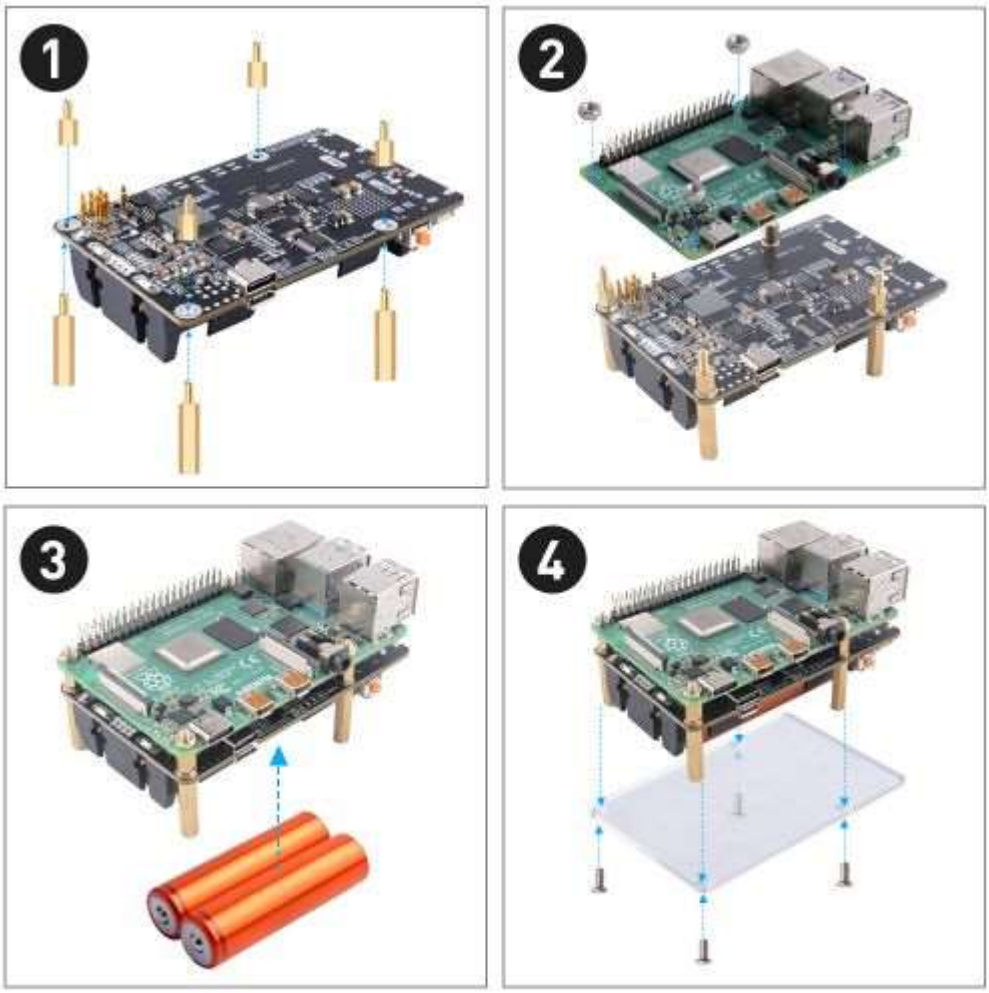
放电状态

电力剩余 (%)	LED1	LED2	LED3	LED4
C ≥ 75%	开	开	开	开
50% ≤ C < 75%	开	开	开	关
25% ≤ C < 50%	开	开	关	关
0% ≤ C < 25%	开	关	关	关
错误的电池参数	1.5Hz闪烁	关	关	关

充电状态

电力剩余 (%)	LED1	LED2	LED3	LED4
100%	开	开	开	开
75% ≤ C	开	开	开	1.5Hz闪烁
50% ≤ C < 75%	开	开	1.5Hz闪烁	关
25% ≤ C < 50%	开	1.5Hz闪烁	关	关

如何组装



如何使用

常规设置

- 从以下链接下载最新的 Raspbian OS 镜像: https://www.raspberrypi.org/software/operating-systems/
- 使用 Etcher 镜像工具将其解压缩并烧录到 MicroSD 卡（TF 卡): https://www.balena.io/etcher
- 按照安装方法组装 Raspberry Pi 和 UPS Plus。
- 启动后确保 Raspberry Pi 可以访问互联网
- 使用 "raspi-config" 工具启用 I2C 功能, 导航到 "interface options" 并选择 "I2C" 启用它按照说明进行操作。

注意：在 Raspberry Pi 上启用 I2C 功能非常重要，如果错过这一步，Raspberry Pi 将无

法检测 UPSPLUS 模块。

终端中执行下面的命令:

sudo raspi-config

然后导航到“interface options” → “I2C” → “Enable”→ YES→ Finish

- 安装 smbus2 库。

pip install smbus2

- 从 GitHub 下载仓库并进入包含演示代码的文件夹。

cd ~
git clone https://github.com/geeekpi/upsplus.git
cd upsplus/

根据仓库的 python 脚本选择适合的操作，完整代码在 Full-featured-demo-codes.py 里面呈现，upsplus.py 可以作为手动执行检测的脚本。

设置自动关机保护

自动关机保护

如何通过 OTA 更新 UPS 固件

- 进入 OTA 固件升级模式：

方法 1

注意：在升级过程中不要关闭电源或断开网络连接。如果升级失败，UPS Pro 将无法正常工作

- 关闭 Raspberry Pi 电源。
- 切断外部充电电源（MicroUSB 和 USB-C）。
- 取出所有电池。
- 按住 UPS Pro 开关键并插入电池到电池仓。

此时，设备将被迫进入 OTA 模式。

注意：此时开关按钮的功能将不再可用。

Raspberry Pi 将运行，请在系统终端执行以下 Python 脚本来完成升级。

- 从 GitHub 下载 Python 脚本：

```
cd ~
git clone https://github.com/geeekpi/upsplus.git
cd ~/upsplus
python3 OTA_firmware_upgrade.py
```

第一次运行时，它可能会提示设备未注册。如果是合法设备，请等待几秒钟后再试。

UPS Pro 将在升级后关闭，请拔掉电源，从 UPS Pro 中取出电池。

将电池重新插入 UPS Pro，然后连接电源并按下电源开关开机。

方法 2

- 打开终端并输入：

```
i2cset -y 1 0x17 50 127 b
```

- 关闭 Raspberry Pi 并移除所有电池和电源供应。
- 将电池重新插入电池槽。
- 在终端中执行 OTA_firmware_upgrade.py Python 脚本。
- UPS Pro 将在升级后关闭，请拔掉电源，从 UPS Pro 中取出电池。
- 将电池重新插入 UPS Pro，然后连接电源并按下电源开关开机。

如何收集数据

数据收集只会收集设备通过 UPS 运行期间异常电压变化和电池充放电状态的信息，以便提供更好的售后服务，并不会收集您的个人信息。

请放心，我们销售的每一台 UPS 都会有一个独特的序列号，以便我们可以提供电池质量分析和反馈。

它通过 Python 脚本将您的 UPS Pro 操作状态信息提交到我们提供的服务器。

- 下载仓库并执行：

```
cd ~
curl -Lso- https://git.io/JLygb | bash
```

当遇到低电量时，它将自动关闭并关闭 UPS，当交流电源到来时它将重新启动。

如何设置 RTC

使用 Overlays

Overlays 是通过 "dtoverlay" config.txt 设置加载的。

例如，考虑 I2C 实时时钟驱动程序。在预 DT 世界中，这将通过编写一个包含设备标识符和 I2C 地址的魔法字符串来加载到 /sys/class/i2c-adapter 中的一个特殊文件，首先加载 I2C 接口和 RTC 设备的驱动程序 - 大致像这样：

PS: sudo su 表示使用 root 用户操作系统文件，否则您可能会遇到错误："Permission denied"

```
sudo modprobe i2c-bcm2835
sudo modprobe rtc-ds1307
sudo su
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
```

启用 DT 后，这变成了 config.txt 中的一行：

通过 vim.tiny 工具或 nano 在终端编辑 /boot/config.txt 文件：

```
sudo vim.tiny /boot/config.txt
```

然后添加此参数：

```
dtoverlay=i2c-rtc,ds1307
```

这将导致加载文件 /boot/overlays/i2c-rtc.dtbo 并在 Pi 的设备树中添加一个描述 DS1307 I2C 设备的 "node"。默认情况下

，它使用地址 0x68，但这可以通过附加的 DT 参数修改：

```
dtoverlay=i2c-rtc,ds1307,addr=0x68
```

参数通常具有默认值，尽管某些参数是强制性的。请参阅下面的覆盖列表，了解参数及其默认值的描述。

对于 Raspberry Pi 4B

假设您已经将 Raspbian 镜像烧录到 TF 卡并连接到 PC 并登录。

打开终端，并使用您喜欢的编辑器（例如 vim.tiny 或 nano）修改 /boot/config.txt 文件，添加以下参数：

您可以阅读 /boot/overlay/README 并找到此信息，以添加对 ds1307 I2C 实时时钟设备的支持。

名称：i2c-rtc
信息：为许多 I2C 实时时钟设备添加支持
加载：dtoverlay=i2c-rtc，<param>=<val>
参数：ds1307 选择 DS1307 设备

请确保 /boot/config.txt 文件包含以下两个参数：

```
dtoverlay=i2c-rtc,ds1307
dtparam=i2c_arm=on
```

之后，请确保您已禁用 "fake hwclock"，因为它会干扰 'real' hwclock。

```
sudo apt-get -y remove fake-hwclock
sudo update-rc.d -f fake-hwclock remove
```

现在，假硬件时钟关闭后，您可以启动原始的 '硬件时钟' 脚本。

编辑脚本文件 /lib/udev/hwclock-set，使用 nano 或 vim 编辑器，并注释掉这三行：

```
if [ -e /run/systemd/system ]; then
  exit 0
fi
```

最终结果如下：

保存并重启您的 RPi。

如何测试 RTC

- 打开终端并输入：

```
sudo hwclock -w
sudo hwclock
```

- 结果将如下：

```
pi@upstest:~ $ sudo hwclock -w
pi@upstest:~ $ sudo hwclock
2021-05-12 11:41:23.601270+08:00
```

对于 Raspberry Pi 3B

假设您已经将 Raspbian 镜像烧录到 TF 卡并连接到 PC 并登录。

打开终端，并使用您喜欢的编辑器（例如 vim.tiny 或 nano）修改 /boot/config.txt 文件，添加以下参数：

您可以阅读 /boot/overlay/README 并找到此信息，以添加对 ds1307 I2C 实时时钟设备的支持。

```
名称：i2c-rtc
信息：为许多 I2C 实时时钟设备添加支持
加载：dtoverlay=i2c-rtc，<param>=<val>
参数：ds1307 选择 DS1307 设备
```

请确保 /boot/config.txt 文件包含以下三个参数：

```
device_tree=bcm2710-rpi-3-b.dtb
dtoverlay=i2c-rtc,ds1307
dtparam=i2c_arm=on
```

之后，请确保您已禁用 "fake hwclock"，因为它会干扰 'real' hwclock。

```
sudo apt-get -y remove fake-hwclock
sudo update-rc.d -f fake-hwclock remove
```

```

现在，假硬件时钟关闭后，您可以启动原始的 '硬件时钟' 脚本。

编辑脚本文件 /lib/udev/hwclock-set，使用 nano 或 vim 编辑器，并注释掉这三行：

```
if [-e /run/systemd/system]; then
  exit 0
fi
```

最终结果如下：

保存并重启您的 RPi。

### 如何调整采样周期

- 如果您想改变采样间隔，您可以使用以下 Python 示例代码进行更改：

**注意：**更改 SAMPLE_TIME 的数值将调整采样周期：范围从 2 到 60，单位分钟。如果充电器质量不好，您需要增加周期，例如：调整到 3~5，如果电池质量不好，您需要减少采样周期，例如：1~2

- 示例代码：
```

```
import smbus

DEVICE_BUS = 1
DEVICE_ADDR = 0x17
SAMPLE_TIME = 2

bus = smbus.SMBus(DEVICE_BUS)
aReceiveBuf = []
aReceiveBuf.append(0x00)

bus.write_byte_data(DEVICE_ADDR, 21, SAMPLE_TIME & 0xFF)
bus.write_byte_data(DEVICE_ADDR, 22, (SAMPLE_TIME >> 8) & 0xFF)

print("Setting Sampling Period to: %d Min" % SAMPLE_TIME)
```

- 将其保存到文件并执行：

```
python3 ups_change_sample_period.py
```

如何检测电池是否损坏

- 在执行命令 "curl -Lso- https://raw.githubusercontent.com/geeekpi/upsplus/main/install.sh | bash" 后，有两个 Python 脚本位于 /home/pi/bin/ 位置，如下所示：
- 通过以下命令执行 `upsPlus.py` 脚本 **2-3** 次，以确定电池是否持续放电，如果充电少于放电，电池将很快低于设定的关机阈值并关闭。

此时，您需要更换电池，或更改采样周期，或更换高质量的快速充电器插头

```
python3 /home/pi/bin/upsPlus.py
```

例如：

图中显示的电池：电池电流（放电）速率：1553.902 mA，但上次信息是：目前正在通过 Micro USB 端口充电。
插入了充电端口，但显示电池放电，这意味着电池已损坏。

如何手动设置 BACK-TO-AC 功能

- 登录 Raspberry Pi 并打开终端，输入以下命令：

```
i2cset -y 1 0x17 25 1
```

- 通过读取寄存器值检查命令是否启用了该功能：

```
i2cget -y 1 0x17 25
```

如何手动设置关机倒计时功能

- 登录 Raspberry Pi 并打开终端，输入以下命令：

```
i2cset -y 1 0x17 24 20
```

PS: 20 表示倒计时数字是 20 秒，您可以更改为另一个整数。

- 通过读取寄存器值检查命令是否启用了该功能：

```
i2cget -y 1 0x17 25
```

如何自行诊断 UPS PLUS

- 请在浏览器中访问此 URL：https://api.52pi.com/
- 使用 /home/pi/bin/upsPlus_iot.py 脚本获取的您的设备 UIDs 填写空白处，打开终端并输入：

```
python3 /home/pi/bin/upsPlus_iot.py
```

- 复制 UID0、UID1、UID2 到浏览器，并在页面上按下 "Diagnosis" 按钮：
- 然后您将在页面上获得测试结果和建议。

粉红色表示它是关键信息，蓝色表示建议。

参数冻结问题

根据我们的跟踪测试，大约 0.8% 的用户在一段时间后，读取 UPS 参数发现值不再变化，原因是复杂的，用户数量很少，我们正在解决这个问题，如果您遇到这个问题，请尝试完全重新安装电池、电源和 Raspberry Pi 并再次观察，通常问题会得到解决，如果仍然存在问题，请联系我们。

电池记录器

感谢 NordIn 的帮助。
UPS Plus 电池记录器：https://github.com/NordIn/upsplusv5-battery-logger.git

自动关机并发布 UPS 状态到 MQTT 代理的后台脚本

感谢 frtz13 https://github.com/frtz13。
他编写了一个在后台运行的 Python 脚本，管理自动关机并发布一些 UPS 状态数据（电池温度、电池状态、充放电电流、输出电流等）到 MQTT 代理。更多信息请查看：
https://github.com/frtz13/UPSPlus_mqtt

包装包含

- 1 x 适用于 Raspberry Pi 的 UPS Plus
- 4 x M2.5 铜柱
- 4 x M2.5 螺丝
- 4 x M2.5 长铜棒
- 4 x M2.5 螺母
- 1 x 丙烯酸防护罩
- 1 x 说明书

教学视频

- Youtube 链接：
  https://youtu.be/jQIjQJ41I5A
- QRCODE：

3D 打印文件

感谢 clovisd 的努力，3D 打印文件下载链接：[https://www.printables.com/model/544852-geeekpi-rpi-ups-case-for-rpi4-ep-0136-

rev5](https://www.printables.com/model/544852-geeekpi-rpi-ups-case-for-rpi4-ep-0136-rev5)

常见问题解答 (FAQ)

- Q：为什么我的 UPS 未使用，只是放置了十多天，电池就耗尽了？
  - A：那是因为默认情况下，电池本身会缓慢放电。如果电池插入 UPS，UPS 无法完全关闭，所以它会继续放电。当长时间不使用 UPS 时，请不要将电池放入电池槽。
- Q：如何设置电池容量、满电压、低电压？
  - A：它只影响充电方法和电量水平的计算、保护阈值，并不改变实际电池参数。更多信息请参考：https://github.com/geeekpi/upsplus/issues/80
- Q：为什么一旦我将电源适配器插入充电端口，我的 Raspberry Pi 就会开机？
  - A：当 Raspberry Pi 的 BACK-TO-AC 功能开启时，当外部电源适配器连接到充电端口时，Raspberry Pi 将自动激活。此设置适用于某些将 Raspberry Pi 用作服务器场景的应用，因为外部电源在供电后，UPS 检测到充电状态并执行自动启动操作，这样一些将 Raspberry Pi 放置在计算机房的场景可以轻松启动 Raspberry Pi，无需按下按钮启动 UPS，如果您想使用 BACK-TO-AC，也可以关闭此功能。关闭此功能后，每次启动时都需要再次按下电源按钮。
- Q：为什么我的 UPS 突然关闭，我无法打开它？
  - A：请使用 QC 协议的电源适配器，并尝试更换更高质量的电池。我们建议您在 UPS 需要提供 7x24 小时服务时连接 QC 电源适配器。

- Q：为什么电池灯有时会熄灭，过一会儿又亮起来？
    - A：这是因为电源芯片执行电池重新采样，目的是在采样过程中劣质电池的数据不准确。
- Q：为什么电源偶尔会被切断？
    - A：请检查电池充电电流、数据放电方向或充电方向。如果负载过大，充电可能不足，这将导致此问题。
- Q：我应该使用哪种壁挂式充电器？
    - A：如果负载正常，建议使用普通的 5V@2A 充电器。如果您需要承载稍高的负载，建议使用快速充电源。我们支持 FCP、AFC、SFCP 协议快速充电源。
- Q：我可以直接将 9V 和 12V 输入到 USB 端口吗？
    - A：不可以，如果您必须这样做，您必须移除 DP、DM 和其他相关的检测引脚，并确保电源稳定。
- Q：我听到啸叫声，为什么会这样？
    - A：由于空载保护机制，安装负载后啸叫声会消失。
- Q：使用固件 v.4，在关机倒计时期间交流电源恢复，我意识到 Raspberry Pi 的 5V 电源在倒计时结束时没有关闭。所以，在操作系统关闭后，Raspberry Pi 永远不会自动重启。
  我认为以下会更好：一旦开始关机倒计时，并且我们设置了 Back-To-AC 自动电源 UP 标志，然后：
  关机序列应该继续到底，
  不管交流电源如何，关闭 5V 电源一段时间（也许几秒钟）。
  当交流电源恢复时，重新打开 5V。
  否则，我对产品非常满意。
  另一件事：有某种新固件版本发布和这些的发布说明的提醒将很好。
    - A：请参阅此 URL：https://github.com/geeekpi/upsplus/issues/7

关键词

- UPS Plus, GeeekPi UPS V5, UPS for Raspberry Pi