# 刷机教程

**https://github.com/a2633063/zDC1**

**仓库，ssh地址：**

**git@github.com:a2633063/zDC1.git**

**拆机与TTL接线方法**

本章节(拆机与TTL接线方法)中所有内容(包含图片)均来自[三木大神的基于esphome的DC1插座自制固件](https://github.com/Samuel-0-0/phicomm_dc1-esphome/tree/master/cookbook),感谢授权使用

image 请勿带市电操作！！！危险自负！！！

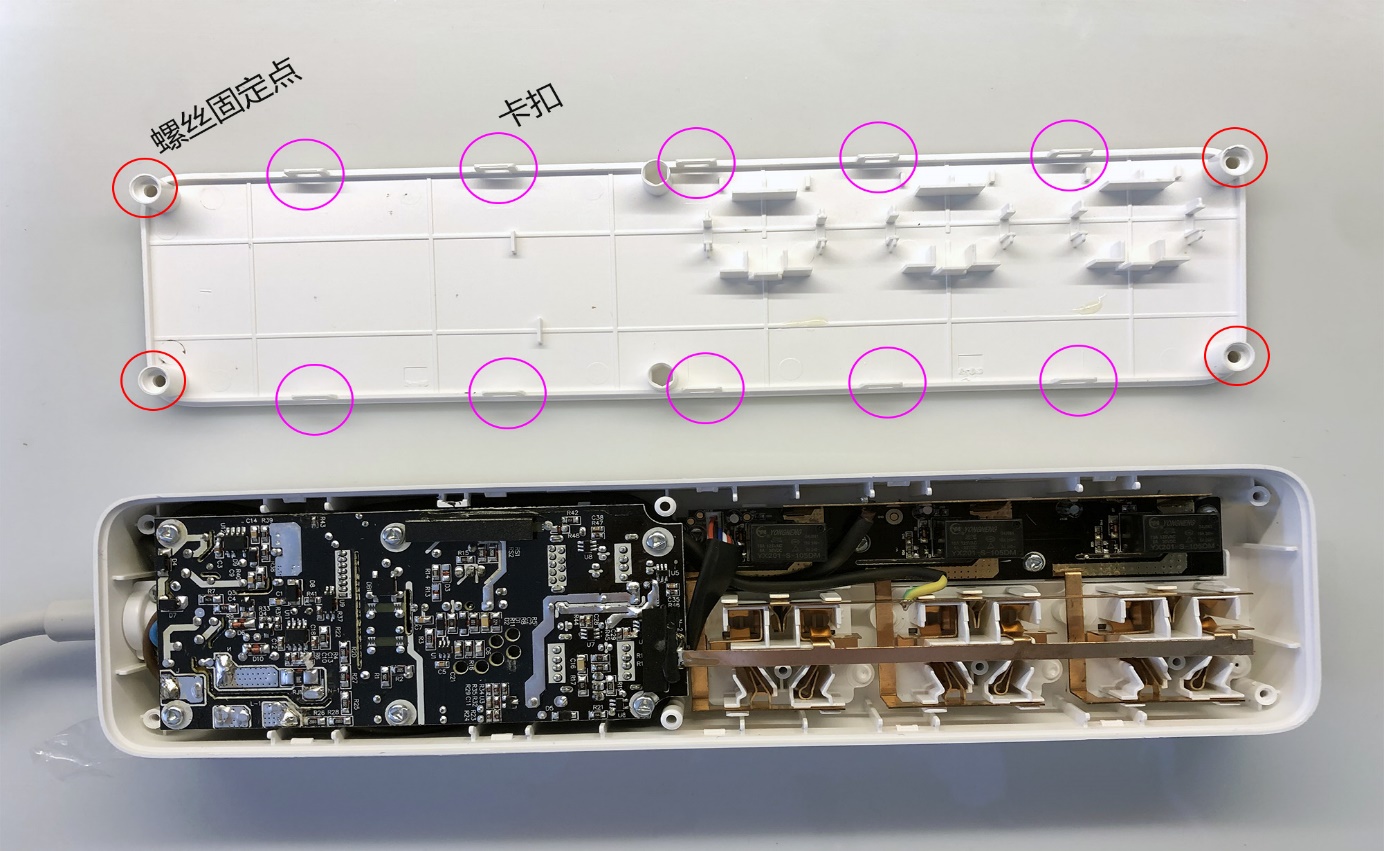
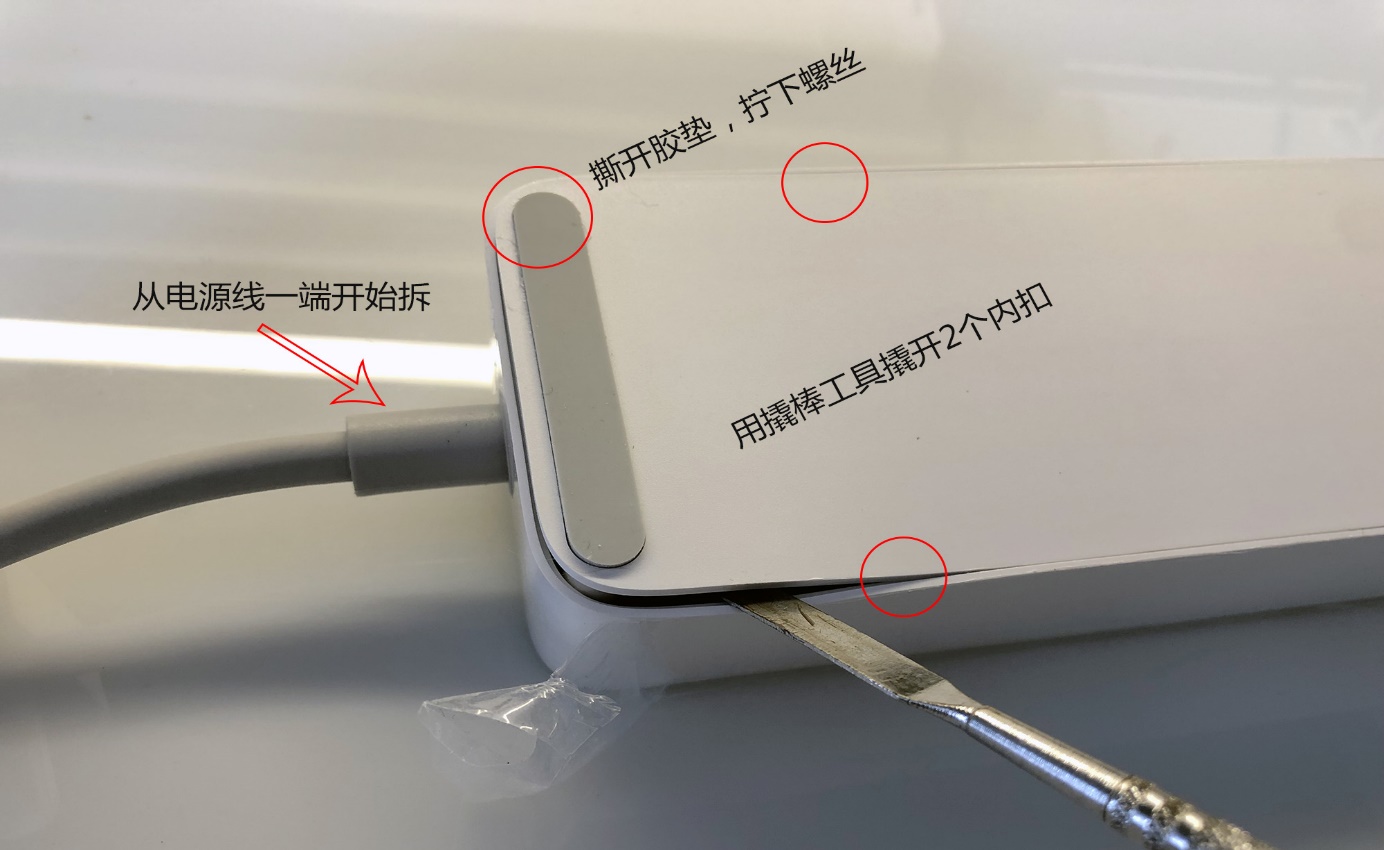
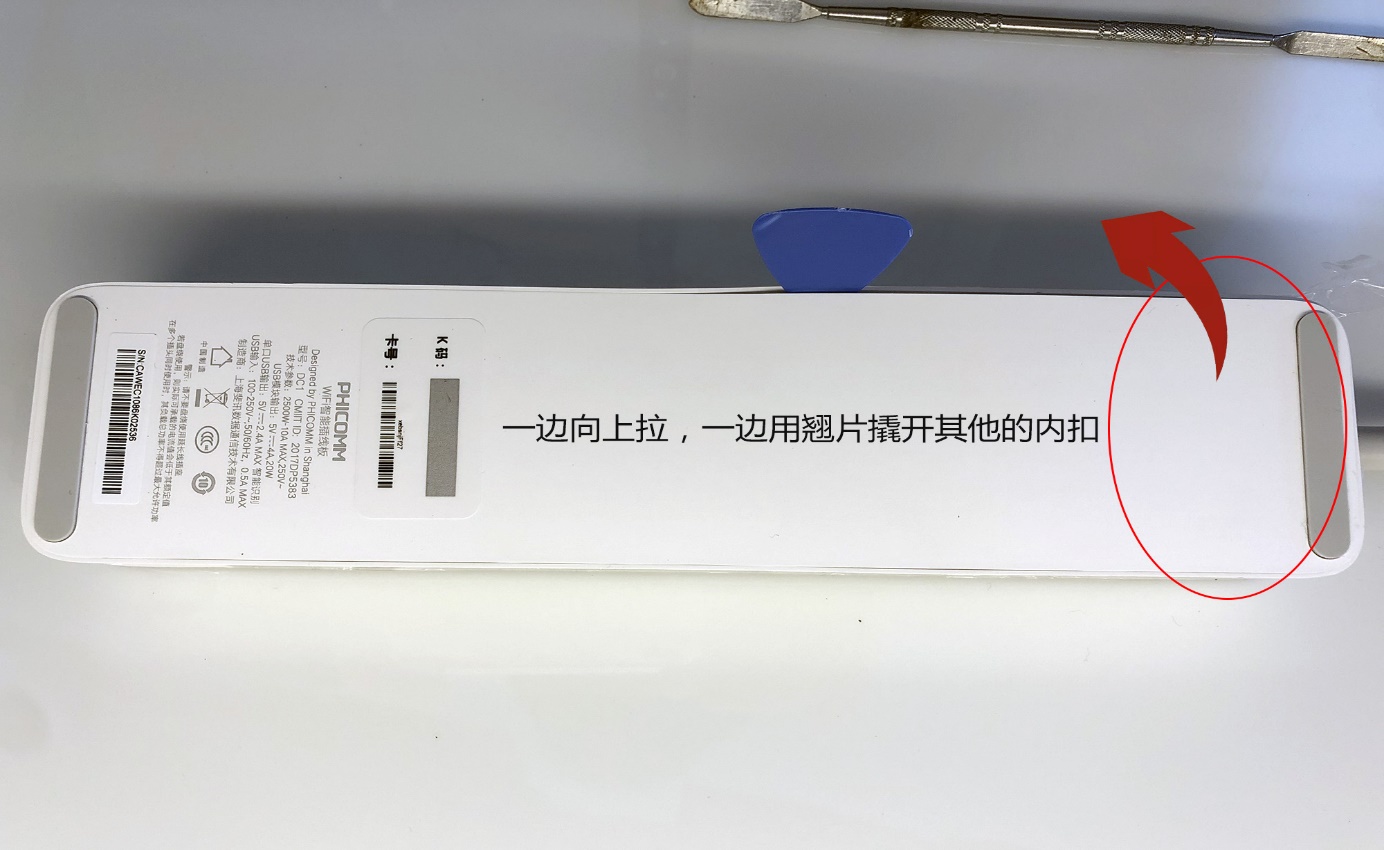
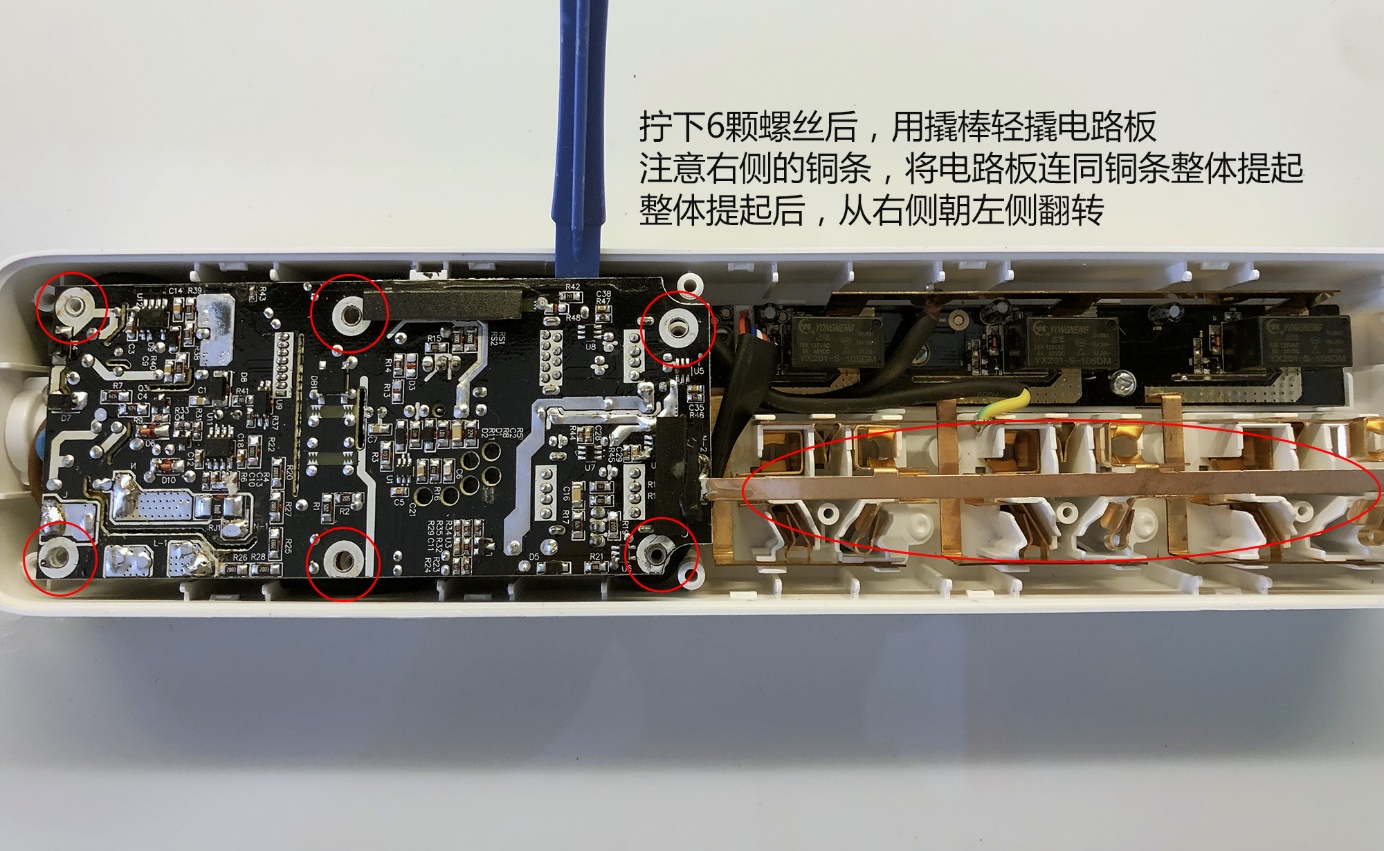
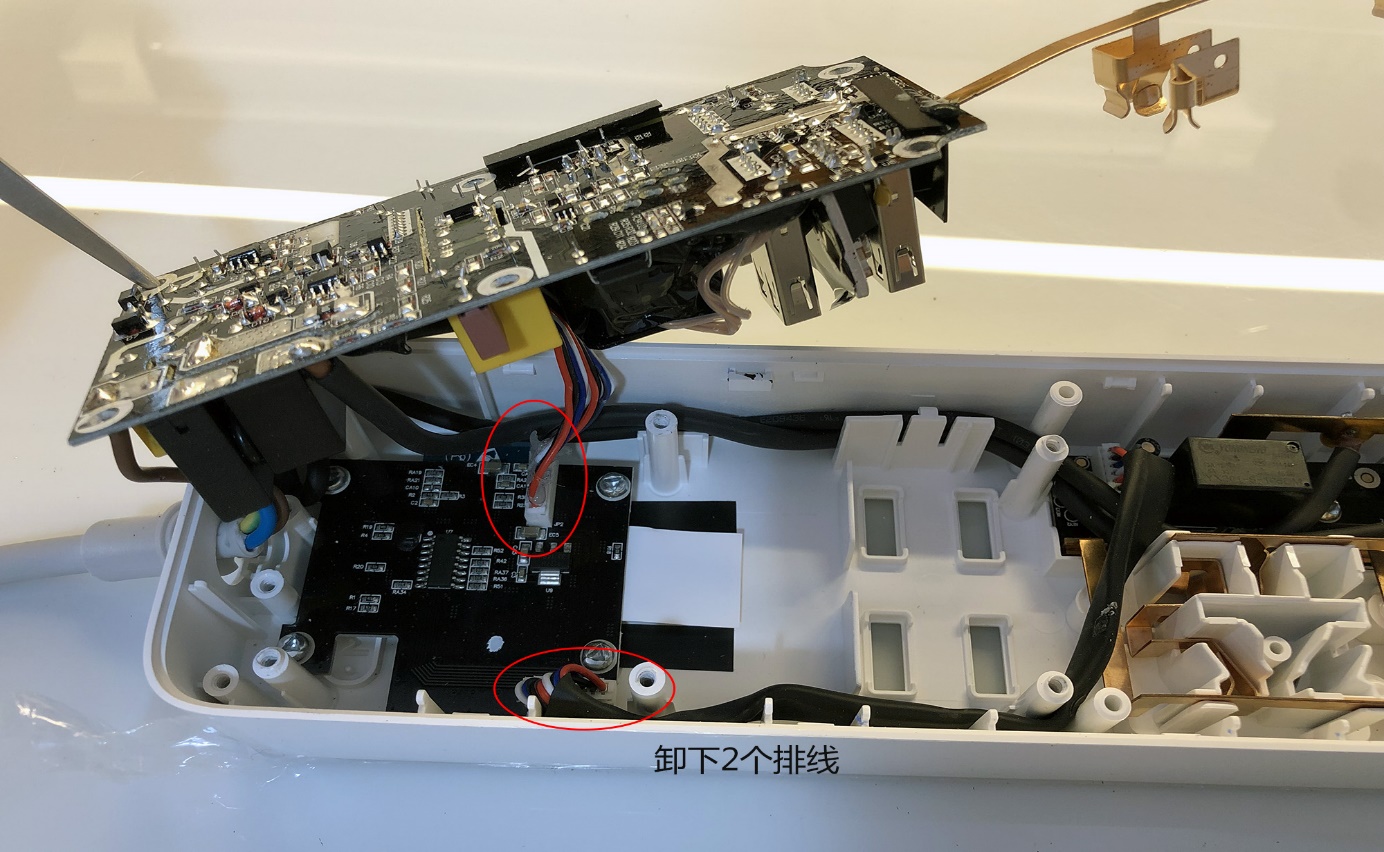
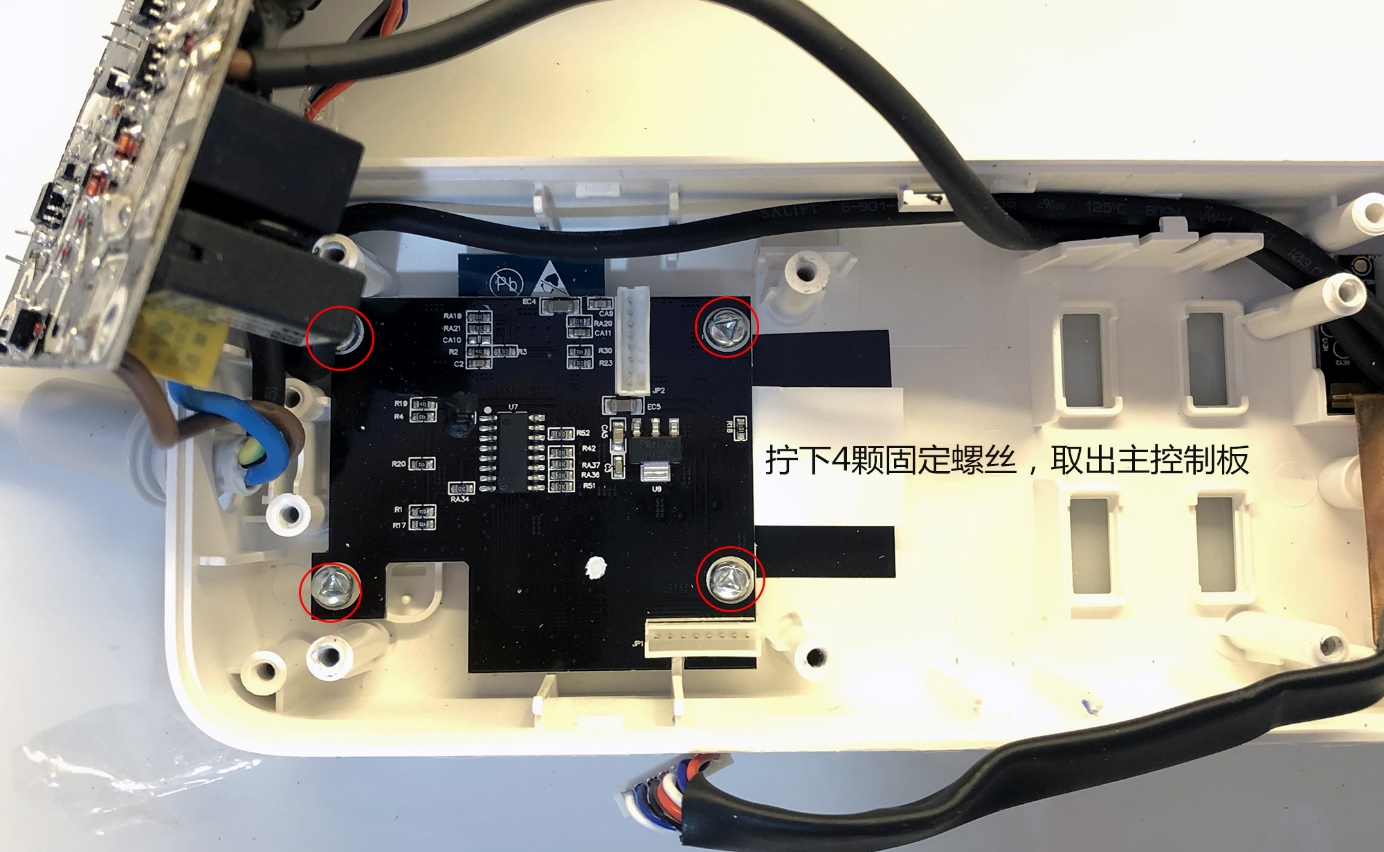
**1、拆主控板**

**用到的工具**



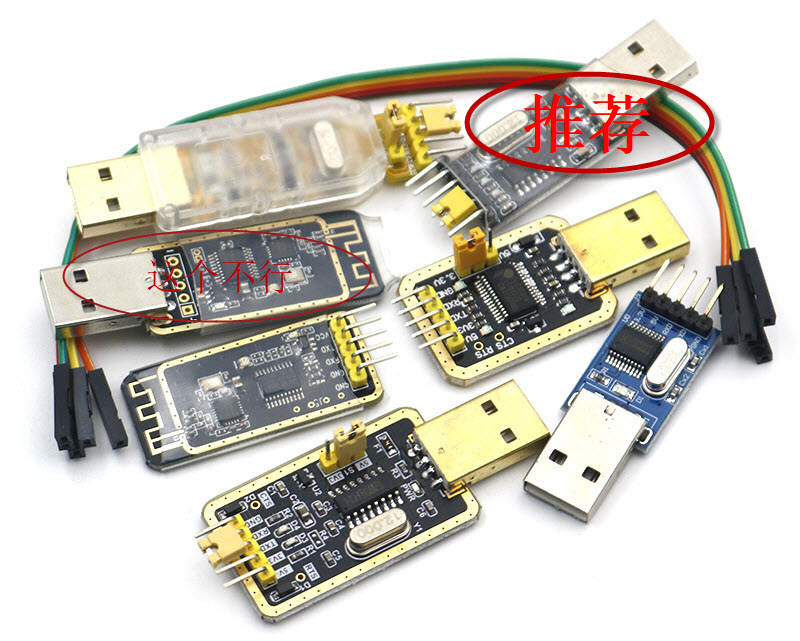
**拆板步骤**

**如果插座刚通过电，拆的时候千万不要去摸电路板，电容带电！！！**

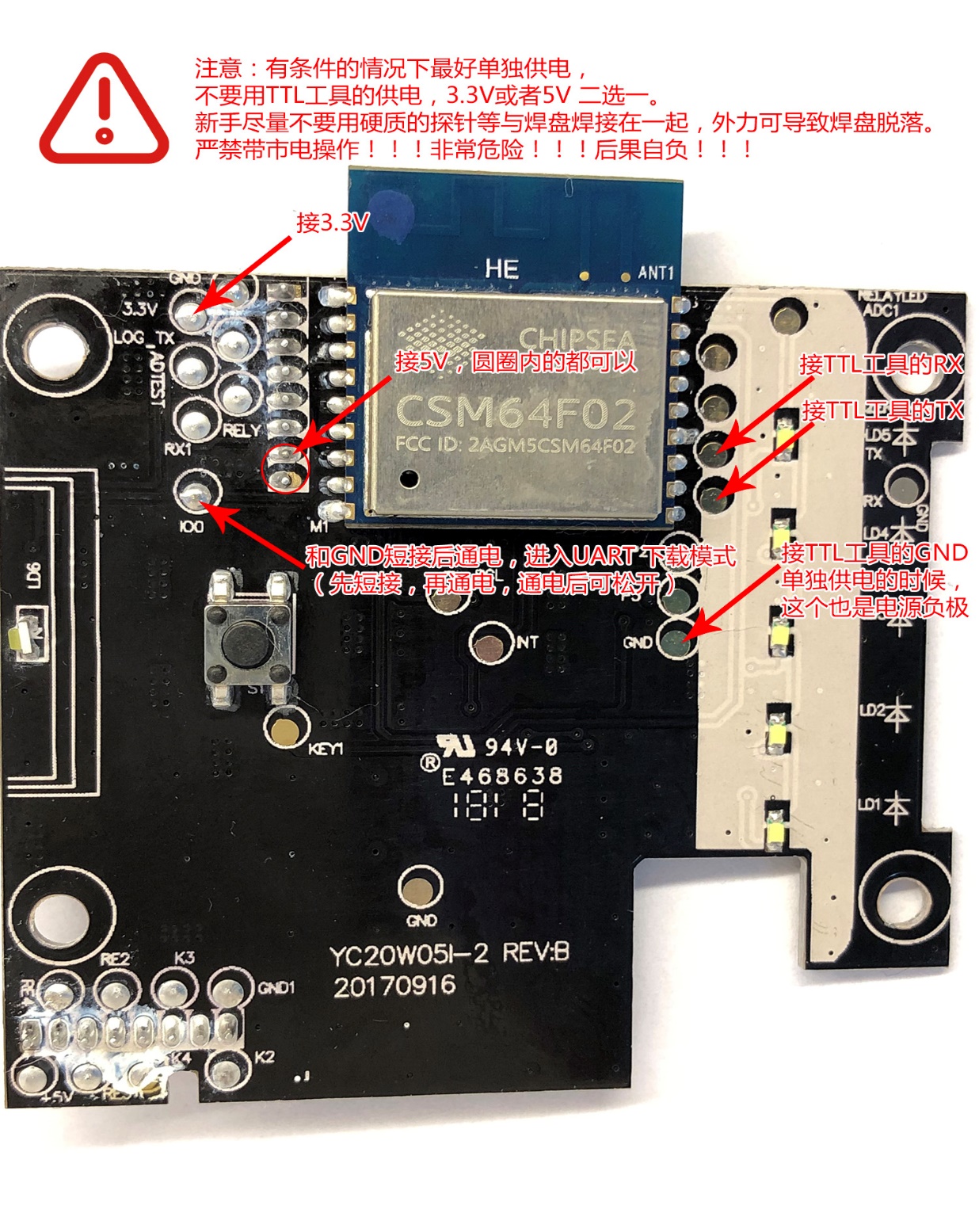
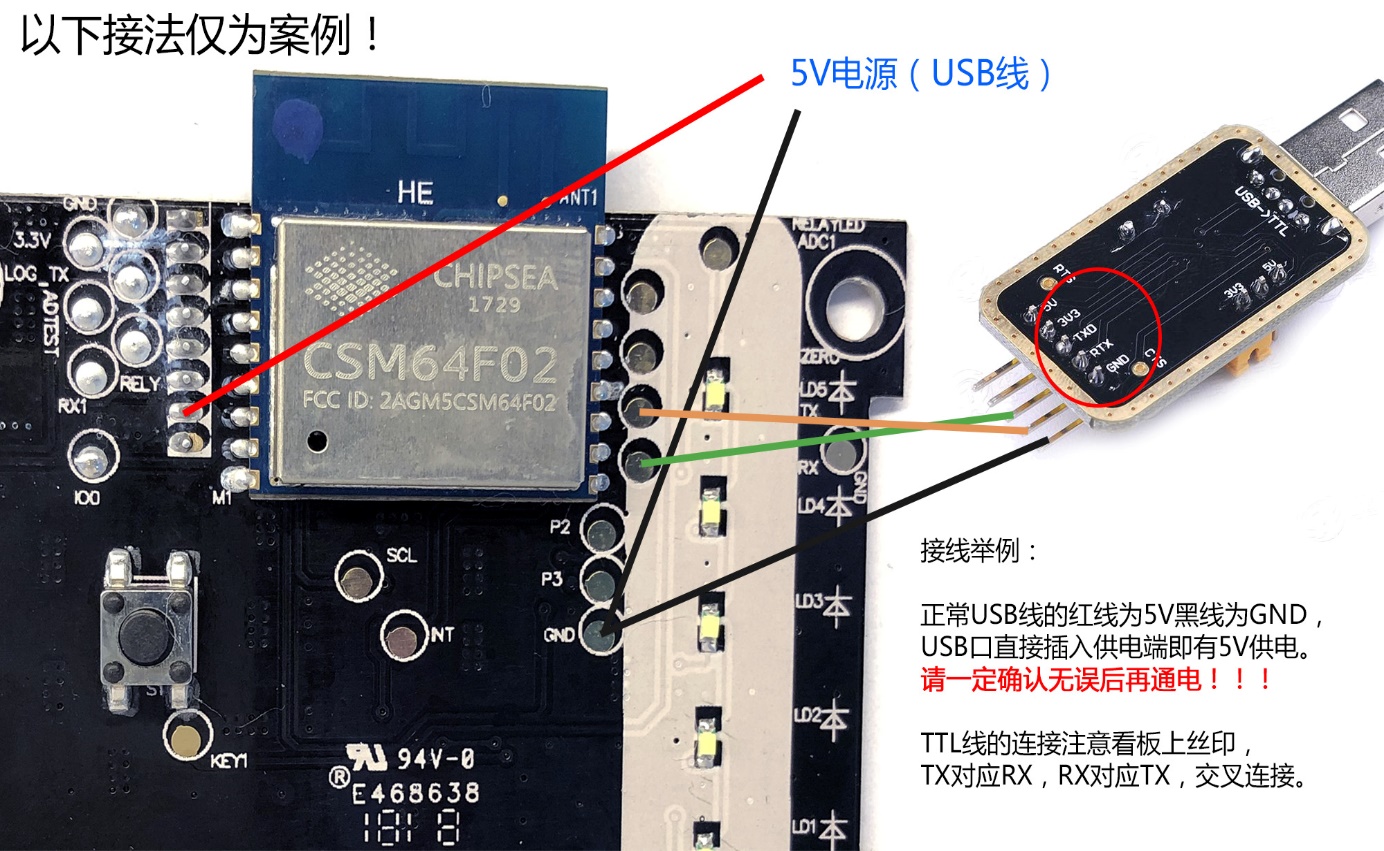
     

**2、主控板接线**

**用到的TTL工具**

任选一个即可 

**接线方法**

**软件刷机**

**工具/固件下载**

确认硬件连接正常后,下载以下软件:

烧录软件: flash\_download\_tools\_v3.6.5.zip [点这里下载](https://github.com/a2633063/zDC1/blob/master/Tool/flash_download_tools_v3.6.5.zip)

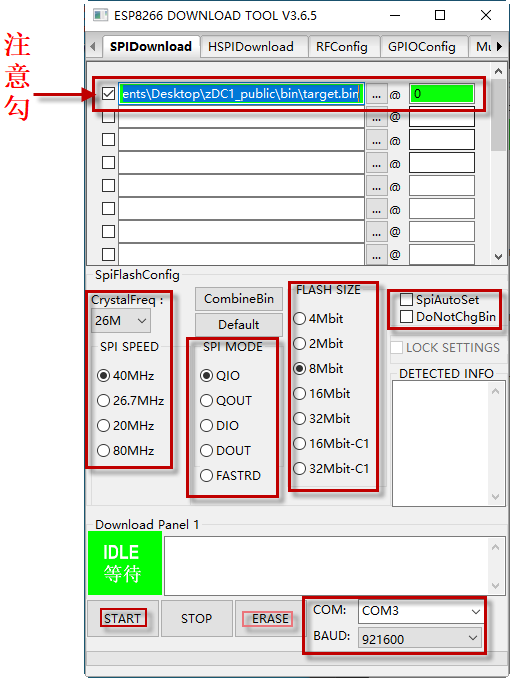
完整固件: target.bin [点这里下载](https://github.com/a2633063/zDC1/releases)

注意:每次release会release出3个文件:user1.1024.new.2.bin、user2.1024.new.2.bin、target.bin

其中，user1.1024.new.2.bin、user2.1024.new.2.bin为ota所需要的文件，target.bin为刷机所用文件，同时targent.bin为user1.1024.new.2.bin与boot\_v1.5.bin文件合并而成.

### 开始烧录

将flash\_download\_tools\_v3.6.5.zip解压,打开目录下的flash\_download\_tools\_v3.6.5.exe,选择ESP8266 DownloadTool,根据以下截图做配置,



将与主控板连接的usbTTL连接上电脑(确保主控io0必需短接gnd后再上电,以进入刷机模式),根据自己的实际串口号设置.,点击START按钮即可开始烧录.

稍等片刻,出现即为烧录超过

注意:部分发现烧录完成后可能出现问题无法使用.可以尝试用以上烧录软件ERASE擦除一次后重新烧录.

进入烧录模式后点ERASE,显示完成即为擦除超过.再将主控板重新上电并再次进入刷机模式,重新点START烧录即可

# 开始使用

此文档对应zDC1固件**v1.0.0及以上**版本.

## 初始化

烧录完成重新开机后,zDC1处于默认设置的状态.所有配置为默认配置.

后续需要恢复默认配置时,可以安住主按键10s直到logo灯及wifi灯同时闪烁后松开按键.zDC1会自动重启并恢复默认配置.

## 配置

### 连接wifi

将排插接入wifi有2种方式:**SmartConfig配置** / **连接排插热点配置**.

2种方式任性其一即可. 建议优先SmartConfig配置. 但是某些刷机过的路由器可能会配置失败,可以使用连接排插热点web配置方式

#### SmartConfig配置

首先需要将设备连接wifi:

1. 手机连接2.4G wifi(待会zDC1需要连接的wifi),打开ZControl app,在侧边栏中单击增加设备按钮,在选择设备类型中选择智能排插zDC1.会自动获取当前wifi名称,手动输入密码.
2. 排插正常上电开机后,按住logo按键不放,等待5秒(如果按按键前logo灯为熄灭,则5此时logo灯会常亮),松开按键.
3. 点击手机app上的配对按钮,等待app弹窗提示配对设备,点击确认增加设备

如果排插已经连上wifi,可以使用获取局域网设备按钮直接获取填加到设备列表中



#### 连接排插热点web配置

若使用SmartConfig失败,可以使用web配网方式.

1. 断开设备电源,按住总开关键上电,大概2秒后松开,连接zDC1\_XXXX热点
2. 打开web:192.168.0.1,输入wifi名称及密码配网.
3. 确认排插已经连上wifi,可以使用获取局域网设备按钮直接获取填加到设备列表中

### MQTT配置

**如果你没有MQTT服务器,甚至不知道MQTT是什么,可以忽略此项设置.**

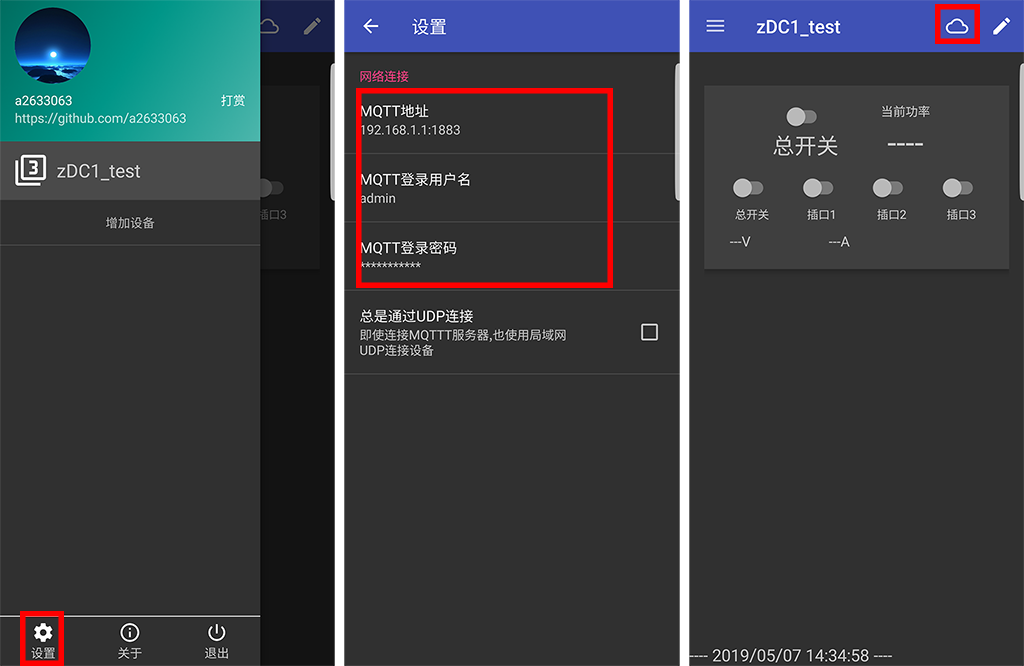
不连接MQTT服务器时,zDC1与手机通过udp通信.

如果你已经配置过mqtt服务器,再次配置时,需要手动重启zDC1才能使新配置生效

配置手机app中的mqtt服务器:

在侧边栏中点击设置按钮,输入MQTT地址,MQTT用户名,MQTT登录密码后返回设备界面.确保底部有提示服务器已经连接.

点击右上角云图标,app会通过udp将MQTT配置发给zDC1.如果正常,zDC1会直接连上MQTT服务器,现在就可以通过MQTT控制设备了.



注意:

​ MQTT地址格式为域名或IP:端口,如地址为www.xyz.com,端口为1883,则应该输入www.xyz.com:1883

## 激活

v1.0.0及以后的版本不再需要激活,直接使用即可.

## 使用

### 设备设置

点击右上角笔形图标,修改设备名称

点击开关图标,控制单个/全部插口通断电.

点击开关图标下的文字,进入单个插口设置页面.

插口页面下点击开关图标,控制此插口通断电.

插口页面下点击开关图标下的文字,设置单个插口此名称.

插口页面下点击定时任务,设置定时任务.

插口页面下点击设置倒计时,设置倒计时任务



注意:名称字符不能超过16个中文,同时,太长时可能会导致名称显示互相覆盖异常.

注意:每个插口能设置5组定时任务,倒计时任务将占用最后一组定时任务,且倒计时任务始终为一次重复.

# homeassistant接入

ZipZhang edited this page on 22 Nov 2020 · [7 revisions](https://github.com/a2633063/zDC1/wiki/homeassistant%E6%8E%A5%E5%85%A5/_history)

此文档对应zDC1固件**v1.0.0及以上**版本.

zDC1支持接入home assistant(一下简称为ha).

## zDC1设置

zDC1通过MQTT服务器接入ha.通过[MQTT配置](https://github.com/a2633063/zDC1/wiki/%E5%BC%80%E5%A7%8B%E4%BD%BF%E7%94%A8#MQTT%E9%85%8D%E7%BD%AE)使zDC1接入ha连接的MQTT服务器.即可

## home assistant配置

注意: 不建议为了此排插入坑hass,使用hass需要很多时间专门来学习相关内容.本人没有精力教授hass相关配置.仅提供配置文件.请自行学习相关配置方式

以下内容中,

MACMAC为设备的mac地址小写,如123456789abc

注意:如果接入多个zDC1,请保证以下 name字段唯一性.(替换MACMAC已经实现此唯一性.若不做其他修改不用管此处)

switch:

- platform: mqtt

name: 'zDC1\_plug0\_MACMAC'

unique\_id: zDC1\_plug0\_MACMAC

state\_topic: 'device/zdc1/MACMAC/state'

command\_topic: 'device/zdc1/MACMAC/set'

payload\_on: '{"mac":"MACMAC","plug\_0":{"on":1},"plug\_1":{"on":1},"plug\_2":{"on":1},"plug\_3":{"on":1}}'

payload\_off: '{"mac":"MACMAC","plug\_0":{"on":0}}'

value\_template: '{{ value\_json.plug\_0.on }}'

state\_on: '1'

state\_off: '0'

# availability\_topic: "device/zdc1/MACMAC/availability"

# payload\_available: 1

# payload\_not\_available: 0

- platform: mqtt

name: 'zDC1\_plug1\_MACMAC'

unique\_id: zDC1\_plug1\_MACMAC

state\_topic: 'device/zdc1/MACMAC/state'

command\_topic: 'device/zdc1/MACMAC/set'

payload\_on: '{"mac":"MACMAC","plug\_1":{"on":1}}'

payload\_off: '{"mac":"MACMAC","plug\_1":{"on":0}}'

value\_template: '{{ value\_json.plug\_1.on }}'

state\_on: '1'

state\_off: '0'

# availability\_topic: "device/zdc1/MACMAC/availability"

# payload\_available: 1

# payload\_not\_available: 0

- platform: mqtt

name: 'zDC1\_plug2\_MACMAC'

unique\_id: zDC1\_plug2\_MACMAC

state\_topic: 'device/zdc1/MACMAC/state'

command\_topic: 'device/zdc1/MACMAC/set'

payload\_on: '{"mac":"MACMAC","plug\_2":{"on":1}}'

payload\_off: '{"mac":"MACMAC","plug\_2":{"on":0}}'

value\_template: '{{ value\_json.plug\_2.on }}'

state\_on: '1'

state\_off: '0'

# availability\_topic: "device/zdc1/MACMAC/availability"

# payload\_available: 1

# payload\_not\_available: 0

- platform: mqtt

name: 'zDC1\_plug3\_MACMAC'

unique\_id: zDC1\_plug3\_MACMAC

state\_topic: 'device/zdc1/MACMAC/state'

command\_topic: 'device/zdc1/MACMAC/set'

payload\_on: '{"mac":"MACMAC","plug\_3":{"on":1}}'

payload\_off: '{"mac":"MACMAC","plug\_3":{"on":0}}'

value\_template: '{{ value\_json.plug\_3.on }}'

state\_on: '1'

state\_off: '0'

# availability\_topic: "device/zdc1/MACMAC/availability"

# payload\_available: 1

# payload\_not\_available: 0

sensor:

- platform: mqtt

name: 'zdc1\_power\_MACMAC'

unique\_id: zdc1\_power\_MACMAC

state\_topic: 'device/zdc1/MACMAC/sensor'

unit\_of\_measurement: 'W'

icon: 'mdi:gauge'

value\_template: '{{ value\_json.power }}'

- platform: mqtt

name: 'zdc1\_current\_MACMAC'

unique\_id: zdc1\_current\_MACMAC

state\_topic: 'device/zdc1/MACMAC/sensor'

unit\_of\_measurement: 'A'

icon: 'mdi:gauge'

value\_template: '{{ value\_json.current}}'

- platform: mqtt

name: 'zdc1\_voltage\_MACMAC'

unique\_id: zdc1\_voltage\_MACMAC

state\_topic: 'device/zdc1/MACMAC/sensor'

unit\_of\_measurement: 'V'

icon: 'mdi:gauge'

value\_template: '{{ value\_json.voltage}}'

homeassistant:

customize:

switch.zDC1\_plug0\_MACMAC:

friendly\_name: zDC1总开关

switch.zDC1\_plug1\_MACMAC:

friendly\_name: zDC1插槽1

switch.zDC1\_plug2\_MACMAC:

friendly\_name: zDC1插槽2

switch.zDC1\_plug3\_MACMAC:

friendly\_name: zDC1插槽3

sensor.zdc1\_power\_MACMAC:

friendly\_name: zDC1功率

sensor.zdc1\_current\_MACMAC:

friendly\_name: zDC1电流

sensor.zdc1\_voltage\_MACMAC:

friendly\_name: zDC1电压

## 其他内容

### 通信协议

此项为专业开发人员准备,如果你不是开发人员,请跳过此项

本项目为防止被低素质人员在我不知情的情况下拿去商用,所以本项目暂时不考虑开源,为了其他开发人员开发自己的控制软件,将通信协议开放

所有通信协议开源,你可以自己开发控制app或ios端

见[通信协议](https://github.com/a2633063/zDC1/wiki/%E9%80%9A%E4%BF%A1%E5%8D%8F%E8%AE%AE)

### FAQ

见 [FAQ](https://github.com/a2633063/SmartControl_Android_MQTT/wiki/FAQ)

zipzhang edited this page on 6 Feb 2020 · [5 revisions](https://github.com/a2633063/zDC1/wiki/%E9%80%9A%E4%BF%A1%E5%8D%8F%E8%AE%AE/_history)

此文档对应zDC1固件**v1.0.0及以上**版本. 此文档还在更新中

**通信格式**

**MQTT主题**

使用MQTT服务器时,

zDC1接收数据监听的topic为device/zdc1/MAC/set,

zDC1发送出的数据的topic为device/zdc1/MAC/state

zDC1定时反馈功率等信息topic为device/zdc1/MAC/sensor

(MAC为小写mac地址字符串,如1234567890ab)

使用UDP时,zDC1接收数据监听的端口为10182,发送数据的端口为10181,发送的目标IP为255.255.255.255,即局域网内全网广播.

通信通过Json格式进行通信,无论是否使用MQTT服务器,通信协议内容相同.

如:

{

"name":"zDC1\_90ab",

"mac":"1234567890ab"

}

当手机app/其他服务器(后面简称为app)通过MQTT或UDP给zDC1设备(后面简称为zDC1或设备)后,zDC1会返回相似的命令.

**控制**

{

"name":"zDC1\_90ab", //当前设备名称,设备依靠此判断此条命令是否应该执行后面的命令

"mac":"1234567890ab", //当前设备mac地址,设备依靠此判断此条命令是否应该执行后面的命令

"ssid":"当前连接ssid", //查询当前连接的wifi名称

"version":"版本信息", //当前固件版本

"setting":

{

"name":"zDC1", //设置zDC1的新名称,最长32个字节

"wifi\_ssid":"你的wifi ssid", //设置连接的wifi ssid

"wifi\_password":"wifi 密码", //设置连接的wifi 密码

"mqtt\_uri":"www.mqtt.com", //设置mqtt服务器

"mqtt\_port":1883, //设置mqtt服务器端口

"mqtt\_user":"z", //设置mqtt服务器登录用户名

"mqtt\_password":"123456" //设置mqtt服务器登录密码

},

"plug\_0": //0-3表示分别表示总开关,1-3插口, 后面的0-3省略

{

"name":"插口1", //当前插口当前名称

"on":1, //当前插口开关

"setting":

{

"name":"新名称", //设置当前插口名称

"task\_0": //设置当前插口定时任务,总共0-4 5组,1-4省略

{ //注意 设置task时,task中所有字段都必须存在而且数据有效

"hour":12 //设置定时任务时间为12:59

"minute":59 //同上

"repeat":10 //重复,bit0-bit6分别表示周一-周日,全部为0时表示一次

"action":1 //动作,0表示关闭插口 1表示开启插口

"on":1 //当前任务是否有效 0表示无效,1表示有效

}

}

}

}

* 以上命令中,"mac"字段为必须有,当有多个zDC1设备时,设备会依靠此字段来判断此条命令是否应该由自己执行.
* mac字段为最根本的识别符,所以无论app发送给设备还是设备发送给app,mac字段都不能省略且不能为null或其他无意义的内容.
* 如果仅仅为了查询某个字段的结果可以将字段冒号后的参数写为null(不带引号,在json中表示空),此时设备不会做任何修改,只会返回当前值
* 请注意以上字段冒号后面携带的参数,有些如1,无引号,表示类型为数字,有些为"名称"表示类型字符.如果类型与要求不同,则与参数写为null效果相同,即为查询.
* 当执行命令和其他功能无关时,其他功能的字段可以省略.
* 设备返回app命令的字符串长度不能超过1023,所以如果发现返回的命令长度可能超过1023,请将命令拆解为多条分别发送.否则可能会引起设备重启的问题.
* 定时任务重复值说明:bit0-bit6分别表示周一-周日,
  + 当全部为0时,即值为0,表示仅一次有效,定时任务执行后,任务会自动变为无效
  + 当全部为1时,即值为127(二进制1111111->十进制127),表每天都有效
  + 当其他情况时,bit0-bit6中哪一位为1,表示每周当天定时任务有效,如值为41时(十进制41->二进制0101001),则bit0 3 5为1,则每周一周四周六定时任务有效.

**功率反馈**

zDC1定时反馈功率电压电流等信息,每秒返回一次.

mqtt 的 topic为device/zdc1/MAC/sensor

{

"mac":"1234567890ab", //唯一标识符

"power":"4.2", //当前功率,单位W

"voltage":"220"， //当前电压,单位V

"current":"0.14" //当前电流,单位A

}

**设备配对**

{"cmd":"device report"},此命令为当app设备列表中无此设备时,发送此命令,设备会返回名称,设备类型编号,mac地址及设备类型名称,

如

{

"name":"zDC1\_1234", //当前设备名称

"type":2, //设备类型编号,2表示zDC1排插

"type\_name":"zDC1", //设备类型名称

"mac":"1234567890ab" //当前设备的mac地址

}

app接收到此命令后,会将此设备加入到列表中.

**ota在线升级**

v0.2版本开始,支持ota在线升级.

查询当前版本:

{

"mac" : "1234567890ab",

"version":null

}

zDC1返回数据:

{

"mac" : "1234567890ab",

"version":"v0.2"

}

v0.2即为当前版本,app可与服务器判断当前最新版本.若当前版本与最新版本不同,则可以发送一下ota升级命令进行升级.

ota升级:

{

"mac" : "1234567890ab",

"setting":

{

"ota1":"http://www.uri.com/user1.1024.new.2.bin",

"ota2":"http://www.uri.com/user2.1024.new.2.bin"

}

}

**dc1主控ota原理为2个固件交替使用(当前为固件1升级后自动切换为固件2,当前为固件2,升级后自动切换为固件1).所以当不确定当前固件为哪个固件时,必须保证ota1 ota2字段都存在且固件下载地址正确!**

其中, "<http://www.uri.com/user1.1024.new.2.bin>" 或 "<http://www.uri.com/user2.1024.new.2.bin>" 为固件下载地址.需要保证用浏览器打开此链接,可以直接下载bin文件即可,且端口必须为80端口.

注意:

链接中不可含有@符号.

链接不可为重定向地址

zDC1返回数据:

{

"mac" : "1234567890ab",

"setting":

{

"ota1":"http://www.uri.com/user1.1024.new.2.bin",

"ota2":"http://www.uri.com/user2.1024.new.2.bin"

}

}

升级完成后,zDC1通过UDP返回数据,并自动重启:

{

"mac":"1234567890ab",

"ota\_progress":100

}

**举例**

app发给设备或设备返回的数据中,字段的顺序没有影响.

**控制mac地址为"1234567890ab"的zDC1设备打开总开关和插口2,关闭插口1.**

app发送zDC1命令(注意总开关、插口1-3编号为plug\_0-3):

{

"mac":"1234567890ab",

"plug\_0":

{

"on":1

},

"plug\_1":

{

"on":0

}

"plug\_2":

{

"on":1

}

}

zDC1返回数据:

{

"mac":"1234567890ab",

"plug\_0":

{

"on":1

},

"plug\_1":

{

"on":0

}

"plug\_2":

{

"on":1

}

}

**设置mac地址为"1234567890ab"的zDC1设备名称为"zdc1"**

app发送

{

"mac":"1234567890ab",

setting:

{

"name":"zdc1"

}

}

zDC1返回:

{

"mac":"1234567890ab",

"name":"zdc1",

setting:

{

"name":"zdc1"

}

}

**查询mac地址为"1234567890ab"的zDC1设备的mqtt服务器**

app发送:

{

"mac":"1234567890ab",

setting:

{

"mqtt\_uri":null

}

}

zDC1返回:

{

"mac":"1234567890ab",

"name":"zdc1",

setting:

{

"mqtt\_uri":"www.mqtt.com"

}

}

**homeassistant 相关协议**

homeassistant同样为读取以上json格式数据来同步/控制状态.