垃圾分类主板与服务器通信协议

--V2.4

一 简述

协议使用

主题命名格式：
设备端发布消息：gc/clientId/pub
设备端订阅消息：gc/clientId/sub

clientId:长度不超过 13 个字节
MQTT 一帧数据的负载长度不超过 450 个字节
发送过程中，字符串之间最好不要有空格，减少传输数据的个数。
发送数据基本格式

基本名词定义:
门：指管理人员收或者放垃圾桶所使用的门，使用电锁控制。
盖：指的是普通用户投放垃圾时打开的投口。

二 JSON 字符串定义

1. 开机时注册服务:

设备端发送: {"id":"clientId","purpose":"request","cmd":"register","content": "clientId"}
服务器回复: {"id":"server","purpose":"reply","cmd":"register","content": "accept/reject"}

开机时会主动向服务器发送数据，服务器需要回应数据，同时在接收到这一帧数据的时候表示主控板上电，这个时候需要配置一些变量，例如定时 IO。

2. 操作垃圾箱

服务器发送:
{"id":"server","purpose":"request","cmd":"operate","content":[{"class":12,"action":"open/close/unlocking"},{"class":13,"action":"open/close/unlocking "}]}
设备端回复:

{"id":"clientId","purpose":"reply","cmd":"operate","content":[{"class":12,"result":"ok/error"},{"class":13,"result":"ok/error"}]}

根据要操作的字符串的长度进行动态的增长，但是总长度是有严格限制的这个要注意

class：为垃圾通分类号，与垃圾桶分控板的分类号保持一致，范围 3~254。

open：打开垃圾箱的盖

close：关闭垃圾箱的盖

unlocking:垃圾箱一般来说都有一个门锁，unlocking 代表打开门锁。

垃圾桶开和关的操作是直接执行的，不会进行开门前的满桶检查。

3. 获取垃圾桶称重数据

服务器发送:

{"id":"server","purpose":"request","cmd":"weight","content":[{"class":12},{"class":13}]}

设备端回复:

{"id":"clientId","purpose":"reply","cmd":"weight","content":[{"class":12,"weight now":12344332,"weight before":12344332,"weight add":12344332,"result":"ok/error"},{"class":12,"weight now":12344332,"weight before":12344332,"weight add":12344332,"result":"ok/error"}]}

重量的单位是 g

可以同时读取多个垃圾桶的数据。

4. 播放音频

服务器发送：

{"id":"server","purpose":"request","cmd":"play audio","content":1}

设备端回复：

{"id":"clientId","purpose":"reply","cmd":"play audio","content":"ok/error"}

5. 设置音量

服务器发送：

{"id":"server","purpose":"request","cmd":"volume","content":1}

设备端回复：

{"id":"clientId","purpose":"reply","cmd":"volume","content":"ok/error"}

音量最大值为 30

6. 垃圾箱状态查询

服务器发送：

{"id":"server","purpose":"request","cmd":"status","content":[{"class":12},{"class":13}]}

设备端回复：

{"id":"clientId","purpose":"reply","cmd":"status","content":[{"class":12,"status":12,"weight":1323232},{"class":13,"status":13,"weight":1323232}]}

"status":是状态数据，直接以数字发送，位定义如下

```
#define STATUS_LIGHT          (1<<0)    //光电
#define STATUS_FULL           (1<<1)    //满桶
#define STATUS_SMOKE          (1<<2)    //烟雾
#define STATUS_DOOR           (1<<3)    //门信号（电锁控制的门）
#define STATUS_OPEN_FAULT     (1<<4)    //开盖故障
#define STATUS_CLOSE_FAULT    (1<<5)    //关盖故障
~~#define STATUS_FULL_INS       (1<<6)    //满桶信号~~
~~#define STATUS_TEST           (1<<7)    //测试信号~~
```

划掉的位和更高位的数据请忽略。

满桶信号和烟雾信号出现的时候，设备会立即向云端发送数据

7. 校准质量

服务器发送：

{"id":"server","purpose":"request","cmd":"weight calib","content":{"class":12, "weight":0}}

设备端回复：

{"id":"clientId","purpose":"reply","cmd":"weight calib","content":{"class":12,"result":"ok/error"}}

校准操作一次只可以操作一个垃圾桶，不同的垃圾桶需要发送多次指令

校准流程，首先发送一个质量为 0，然后设置质量为一个固定的数进行校准。

也可单独写质量 0，此时可以认为是去皮操作，将当前质量设置为 0.

这条指令 服务器需要等待较长时间 因为主控需要先与分控板进行通信才可以发送数据给服务器。

8. 质量校准系数读取

服务器发送：
{"id":"server","purpose":"request","cmd":"weight fread","content":{"class":12}}
设备回复：
{"id":"clientId","purpose":"reply","cmd":"weight fread","content":{"class":12,"gain":1.001,"offset":2}}
一次只可以操作 1 设备

9. 质量校准系数写入

服务器发送：
{"id":"server","purpose":"request","cmd":"weight fset","content":{"class":12,"gain":1.001,"offset":2}}
设备回复：
{"id":"clientId","purpose":"reply","cmd":"weight fset","content":{"class":12,"result":"ok/error" }}

10. 读 GPS 坐标

服务器发送：
{"id":"server","purpose":"request","cmd":"rGPS","content":"readGPS"}
设备端回复：
{"id":"clientId","purpose":"reply","cmd":"rGPS","content":{"longitude":"12.23424334343","latitude":"13.23424334343"}}

11. 写 GPS 坐标

设备没有 GPS 芯片，需要在安装时使用其他设备写入一个固定的 GPS 地址。

服务器发送：
{"id":"server","purpose":"request","cmd":"wGPS","content":{"longitude":"12.23424334343","latitude":"13.23424334343"}}
设备端回复：
{"id":"clientId","purpose":"reply","cmd":"wGPS","content":"ok/error"}

12. 垃圾箱状态发送

设备端定期发送给服务器，如果没有其他数据，则 10 分钟会发送一次，如果在 10 分钟内发生了通信行为，则心跳不会发送。当某些状态发生改变时，例如垃圾桶满，会立即发送。

设备端发送：

{"id":"clientId","purpose":"request","cmd":"status report","content":[{"class":13,"status":12,"weight":1323232},{"class":13,"status":13,"weight":1323232}]}

服务器回复:

{"id":"server","purpose":"reply","cmd":"status report","content":[{"result":"ok "},{"result":"ok "}]}

"status"的定义见 6。

13. 客户卡信息验证

设备端发送：

{"id":"clientId","purpose":"request","cmd":"card check","content":12345344}

服务器回复：

{"id":"server","purpose":"reply","cmd":"card check","content":"accept/reject/ok"}

14. 客户二维码信息验证

设备端安装有二维码的读头，扫描客户提供的二维码。

设备端发送：

{"id":"clientId","purpose":"request","cmd":"QR check","content":"http://xxxxxxxx"}

服务器回复：

{"id":"server","purpose":"reply","cmd":"QR check","content":{"user id":123344532,"result":"accept/reject/ok"}}

此处需要返回"user id"，此 id 是服务器端用来标识用户的标志位。垃圾箱在操作过程中会返回状态，通过此 user id 来分辨是哪个用户的数据。

15. 执行开箱序列（开盖）

使用此指令时，主控板会自动开关盖，上传质量，播放语音，推荐使用。

服务器发送：

{"id":"server","purpose":"request","cmd":"seq open","content":{"class":12, "user id":123344532,"action":"open/close"}}

设备端回复：

{"id":"clientId","purpose":"reply","cmd":"seq open","content":{"class":12, "user id":123344532,"result":"ok/error"}}

一次只可以操作一个垃圾桶。

16. 开箱序列执行状态

设备端发送：

{"id":"clientId","purpose":"request","cmd":"seq status","content":{"class":12, "user id":123344532,"trigger":"button/card/QR/server/usb","action":"open/opening/close/closing","weig ht now":12344332,"weight before":12344332,"weight add":12344332}}

服务器一共可有收到 4 帧数据，只有在 action 为 close 的时候，后面的质量才是有效的。

服务器回复：

{"id":"server","purpose":"reply","cmd":"seq status","content":"ok"}

17. 主控功能读取指令

服务器发送：

{"id":"server","purpose":"request","cmd":"main read","content":"ok "}

time：开门的超时时间，单位秒，整数，不支持小数

设备端回复

{"id":"clientId","purpose":"reply","cmd":"main read","content":123442}

content 的内容待定

18. 主控功能配置指令

服务器发送：

{"id":"server","purpose":"request","cmd":"main set","content":12344}

time：开门的超时时间，单位秒，整数，不支持小数

设备端回复

{"id":"clientId","purpose":"reply","cmd":"main set","content":"ok/error"}

content 的内容待定

19. 分类功能读取指令

服务器发送:

{"id":"server","purpose":"request","cmd":"sub read","content":"ok"}

设备端回复:

{"id":"clientId","purpose":" reply ","cmd":"sub read","content":[{"class":12,"time":6, "config":3},{"class":13," time ":6, "config":2}]}

20. 分类功能配置指令

服务器发送：

{"id":"server","purpose":"request","cmd":"sub set","content":{"class":12,"time":10,"config":1}}

设备端返回：

{"id":"clientId","purpose":"reply","cmd":"sub set","content":{"class":12, "result":"ok/error"}}

只能单个操作，不能一条指令操作多个

需要等待时间长一些。

21. 分控板温湿度读取指令

服务器发送：

{"id":"server","purpose":"request","cmd":"humiture","content":[{"class":12},{"class":13}]}

要读取几个分类 就读取就写几个类别。

设备端返回：

{"id":"clientId","purpose":"reply","cmd":"humiture","content":[{"class":12,"temp":2344, "hum":2344},{"class":13,"temp":2344, "hum":2344}]}

temp:温度 范围 0~4095

hum :湿度 范围 0~4095

22. IO 写指令

服务器发送：

{"id":"server","purpose":"request","cmd":"IOW","content":"000001"}

设备端返回：

{"id":"clientId","purpose":"reply","cmd":"IOW","content":"000001"}

说明：

content：000011　代表：IO7~IO9 与 Q10~Q12 的输出状态，板上的芯片可以更改配置，然后将所有 IO7~IO9 都设置成为输出。默认 Q10~Q12 是输出，这里直接兼容了所有情况。

如果希望 Q12 输出为高：content 写 "000001"

如果希望 Q11，Q12 输出为高：content 写 "000011"

23. IO 读指令

服务器发送：

{"id":"server","purpose":"request","cmd":"IOR","content":"ok"}

"content"这个标签可以不写，主控板不关心。

设备端返回：

{"id":"clientId","purpose":"reply","cmd":"IOR","content":"000000000001"}

此条指令回报 12 个 IO 的情况。

在没有接设备的情况下，输入 IO 默认读到的信号是高电平，即是 1.

24. 定时 IO

服务器发送：

{"id":"server","purpose":"request","cmd":"timeIO","content":{"io":7, "time":"103100-110101-5000-1000"}}

设备端返回：

{"id":"clientId","purpose":"reply","cmd":"timeIO","content":"ok/error"}

content 内部的配置：

"content":{"io":7, "time":"103100-150101-5000-1000"}

表示配置 io7 ，有效时间是从每天的 10 点 31 分 00 秒到 15 点 01 分 01 秒，间隔时间 50 分 00 秒，消毒时间 10 分 00 秒

"content":{"io":12, "time":"230101-070202-5000-1000"}

表示配置 io12，开启时间从每天晚上 23 点 01 分 01 秒到次日凌晨 7 点 02 分 02 秒，间隔时间 50 分 00 秒，消毒时间 10 分 00 秒

注意时间的设置，即使间隔时间很短只有几秒钟，也必须写完整，否则会报错，例如：

"time":"103100-110101-0009-0009"，9 秒之前的 0 也是不可省略的。

将 time 的起始时间和结束时间都设置成为 0，表示关闭这个功能，例如：

"time":"000000-000000-0000-0000"