工业物联网协议
（wlink TCP）

www.ayi9.com

咨询电话 13701036459（微信同名）

2021.01.20 修订

0 目录

1 物联网协议

互联网正在应用到各个方面,它正在向传统工业的渗透.我们来讲讲工业物联网.

工业物联网就是把传统的工业设备接入互联网. 大到机床小到一个简单的工具或者设备, 例如不间断电源, 或传感器,如温度计,湿度计,PM2.5检测器等.

要跟这些设备对话,必须有一套交流语言。 这就是我们要讲的**工业物联网协议**

我们使用 json 格式来表达通讯的数据.

```
{
  "name":"ayi9 company",
  "addr":"广东深圳"
}
```

我们将使用下面术语

- 设备 Device : 指具体的设备,例如 电源, 温度计, 湿度计等. 设备充当**服务者**的角色
- 客户机 Client : 可能是手机 app 或 电脑软件, 它获取设备的数据和控制设备. 它是**设备**的**客户**

我们只讨论 ip 网络. 其它非 ip 设备可以通过 ip 设备代理它.

2 发现设备

每个设备都有

```
uid : 设备编号 例如 "123456@ayi9.com"
type : 设备类型 例如 "ups@ayi9.com"
```

设备编号

设备编号 通常采用 123456001@domain 这样的形式.

设备类型

设备类型 通常采用 type@domain 这样的形式.

向局域网广播查找

要跟设备对话, 首先要找到它. 最简单的是查找局域网内的设备, 使用下面的办法

向局域网广播地址 udp:8100 广播查找信息

```
{
  "search":1,
  "seq":10
}
```

或 指定要查找的设备编号

```
{
  "search":1,
  "uid":"12345678@ayi9.com",
  "seq":10
}
```

设备在 udp:8100 侦听到这个查询,它会回复如下

```
{
  "report":1,
  "uid":"12345678@ayi9.com",
  "type":"ups@ayi9.com",
  "name":"IDC ups",
  "seq":10
}
```

广播查找的格式是这样的

```
{
  "search":<版本号>, 当前版本号为 1
  "uid":"<设备编号>", 这个是可选项
  "type":"<设备类型>",这个是可选项
  "seq":<报文编号>
}
```

如果查询中指定了 uid, type 可选项, 只有那些符合条件的设备才会给出回复.

设备回复格式是这样的

```
{
  "report":<版本号>, 当前版本号为 1
  "uid":"<设备编号>", 例如 "123456@ayi9.com"
  "type":"<设备类型>",例如 "ups@ayi9.com"
  "name":"<设备名称>",例如 "深圳机房 UPS 101"
  "seq":<报文编号> 对应查询的报文编号
}
```

设备回复中可能还有其它字段, 如果你认识它们的话就可以利用. 不认识的话, 当它没有.

```
{
  "report":1,
  "uid":"12345678@ayi9.com",
  "type":"ups@ayi9.com",
  "name":"IDC ups",
  "brand":"STK" ,
  "model":"C3K" ,
  "seq":10
}
```

查询设备的目的是找到它的 ip 地址.就是它的回复包的源地址.

3 连接设备

我们通过某种途径(例如上节所述广播查询的办法)已经了解到设备的 ip 地址, 设备在 TCP:8200 提供物联网服务. 一次 TCP 连接就是一个会话.

在会话中,请求和回复都是一个完整的 json 结构**{数据包}**. 对于**{数据包}**之间的(空白)字符应该被忽略.

大多数设备端程序支持一问一答。也就是说没有获得回应之前不要发出新的请求。 这个要求跟 FTP, HTTP 是一样的。

首先向它发送连接请求.

```
{
  "connect":1,
  "pass" : "123456"
}
```

连接请求的基本格式是

```
{
  "connect":<连接者身份>, 很多情况下用数字表示身份。
  "pass":"<连接密码>" 这个是可选项
}
```

也可以用字符串表示身份，例如

```
{
  "connect":"myname"
  "pass" : "123456"
}
```

正常情况下, 设备会回应

```
{
  "answer": "200 welcome"
}
```

也有可能得到

```
{
  "answer": "505 busy now"
}
```

如果设置了 pass, 有可能得到

```
{
  "answer": "405 auth error",
  "challenge": "qjud3w1jxs"
}
```

如果没有设置 pass, 有可能得到

```
{
  "answer": "404 auth required",
  "challenge": "qjud3w1jxs"
}
```

这个 challenge 用于密码认证. 称之为 auth_challenge.

你需要加上认证码 再连接

```
{
  "connect":1,
  "auth":"bacde8574566ef...."
}
```

这里的认证码

```
auth = hex( sha( sha(pass) + auth_challenge ) )
```

如果认证码错误, 设备应答

```
{
  "answer": "405 auth error",
  "challenge": "qjud3w1jxs"
}
```

如果认证通过, 你得到 200 回应, 那么你可以正式跟设备对话了.

{ 现在是对话时间 下一节细讲 }

对话结束后, 你可以礼貌地离开而不是直接关闭 socket 走人.

```
{
  "bye": "bye"
}
```

设备回复

```
{
  "answer": "202 bye"
}
```

4 对话设备

在上一节我们搞清楚了如何跟设备建立起连接. 本节我们讲解如果获取数据,设置数据和执行命令.

机内每个数据项都有名称.

使用 get 获取数据

获取名称为 brand 的数据

```
{
  "get":["brand"]
}
```

回复长这样（关键字段是 "return"）

```
{
  "return":["STK"]
}
```

获取数据的基本格式是

```
{
  "get":["<数据名称>"]
}
```

回复的基本格式是

```
{
  "return":[<数据>]
}
```

可以同时获取多项数据

```
{
  "get":["<数据名称1>","<数据名称2>","<数据名称3>,,,"]
}
```

回复的格式是

```
{
  "return":[<数据1>,<数据2>,<数据3>,,,"]
}
```

这里的**<数据>**,可以是简单的字符串,数字,或者复杂的结构化数据.例如,

```
  "brand" : "STK"
  "network" : [ "192.168.0.220",24,"192.168.0.1",
                  "192.168.0.1" ]
```

如果获取的数据项不存在,其返回的**数据**是 **false**, 例如

```
{
  "get":["brand","other"]
}
```

回复长这样

```
{
  "return":["STK",false]
}
```

设备可以获取什么样的数据,完全与设备类型(type)有关.例如,电源和温度计可以读取的数据显然是不同的.

但是,所有设备都应该有 info 数据, 它至少含有下面内容

```
{
  "type":"<设备类型>"
}
```

例如

```
{
  "get":["info"]
}
```

回复可以是

```
{
  "return":[{"type":"ups@ayi9.com","name":"ONE"}]
}
```

客户机可以根据 info 中的 type 发出正确的请求。

使用 set 设置数据

设置名称为 name 的数据

```
{
  "set":{"name":"TEST ONE"}
}
```

回复长这样 (关键字段是 "return")

```
{
  "return":[true]
}
```

设置数据的基本格式是

```
```

```
{
  "set":{"<数据名称>":<数据>}
}
```

回复的基本格式是

```
{
  "return":[true]
}
```

可以同时设置多项数据

```
{
  "set":{"<数据名称1>":<数据1>,"<数据名称2>":<数据2>,,,}
}
```

回复的格式是

```
{
  "return":[true,true,true,,,]
}
```

如果设置的数据项不存在或者不允许设置,返回 **false**, 例如

```
{
  "set":{"name":"TEST", "other":30}
}
```

回复长这样

```
{
  "return":[true,false]
}
```

设备可以设置什么样的数据,完全与设备类型有关.

使用 cmd 执行命令

可以要求设备执行命令

```
{
  "cmd":["switch on"]
}
```

回复长这样 (关键字段是 "return")

```
{
  "return":[true]
}
```

执行命令的基本格式是

```
{
  "cmd":["<命令>"]
```

```
}
```

回复的基本格式是

```
{
  "return":[<命令回复>] 经常是简单的 true
}
```

可以同时执行多条命令

```
{
  "cmd":["<命令1>","<命令2>","<命令3>",,,]
}
```

回复的格式是

```
{
  "return":[<命令回复1>,<命令回复2>,<命令回复3>,,,]
}
```

如果命令执行失败,返回 **fail**, 如果命令不存在,返回 **false**, 例如

```
{
  "cmd":["escape"]
}
```

回复长这样

```
{
  "return":[false]
}
```

设备可以执行什么样的命令,完全与设备类型有关.

5 对话小结

前面两节我们介绍了如何连接, 如何对话. 它们是

```
  { "connect":,,,, }
  { "bye":"bye" }
  { "get":[,,,],,, }
  { "set":{,,,},,, }
  { "cmd":[,,,],,, }
```

他们的回应是

```
  { "answer":"..." } 或
  { "return":[,,,] }
```

如果你不使用上面的请求格式, 或设备不理解你的请求, 回应将是

```
{
  "answer":"410 bad request"
}
```

answer 回应的内容有

```
200 OK
202 bye
404 auth required
405 auth error
410 bad request
505 busy now
```

6 UPS 协议

ayi9 公司在 UPS，稳压器，工业空调，电热器, 电梯等许多设备上都在使用物联网协议。

对于每种设备，我们定义它的设备类型和它可以 get, set, cmd 的内容(称之为设备协议)

对于 UPS, 我们使用了 ups@ayi9.com 作为它的类型。下面是它可以操作的部分内容.

connect

例如 { "connect":1, "pass":"123abc" }
或者 { "connect":1, "auth":"abcdef00..." }

get

info	设备信息 [type,,,,,]
RTD	额定信息 [品牌,型号,硬件版本,输入电压,输出电压,输出电流,输出频率,电池电压,输入相位/输出相位,功率]
QAD	动态信息 [输入电压[3],输入频率,输入电流[3],故障电压,输出电压[3],输出频率,输出电流[3],负载百分比[3], 电池电压，电池剩余时间（分钟）,电池容量百分比,机内温度,状态]

set

name	名称
pass	密码

cmd

SON	开机
SOFF	关机
T	10秒放电测试 自动结束
TL	开始放电测试
CT	结束放电测试
BZON	蜂鸣允许
BZOFF	蜂鸣禁止