Contents

1 ChirpStack 网关 API

1.1 从 MQTT broker 订阅消息

version 0.11

`application/2/device/32722cdd1c277953/rx`

```
{
    "applicationID": "2",        //应用 ID，没用，目前暂定为 2
    "applicationName": "app",    //名称，没用
    "deviceName": "811",         //设备名
    "devEUI": "32722cdd1c277953", // 设备 ID，在 topic 的 device 下
    "rxInfo": [{
        "gatewayID": "b827ebfffecce597", // 网关 ID
        "uplinkID": "78b71b9c-05e3-4feb-9aa9-8b1dbcd745a7", // 上行 ID，没用
        "name": "rak-gateway", // 网关名称
        "rssi": -58, //信号强度，有用
        "loRaSNR": 9.2, // LoRa 的信噪比
        "location": { // 网关的位置，不是终端节点的位置，那是在二进制数据里面
            "latitude": 24.93545,
            "longitude": 118.64048,
            "altitude": 18
        }
    }],
    "txInfo": {
        "frequency": 487100000, // 传输所在频率
        "dr": 5 // 数据率
    },
    "adr": true, // 是否启用动态数据率功能
    "fCnt": 0, //下行计数器，连续发送的消息的话计数器也应该是连续的，没啥用
    "fPort": 1, // 端口，确定消息类型
    "data": "WgA=", //经过 base 64 编码的二进制数据，储存这个就好了
    "object": {
        "DecodeDataHex": "0x5a,0x00", // 解码后的每个 byte
```

```
        "DecodeDataString": "Z\u0000" // 根据 ASCII 或者 UFT-8的译码字符串
    }
}
```

1.2 发布消息到 MQTT broker

```
mosquitto_pub -h example.com -t "application/2/device/32722cdd1c277953/tx" -f test.json -d
```

```
{
    "confirmed": true,
    "fPort": 10,
    "data": "YWFh" //base64 encoded aaa
}
```

1.3 利用 RESTful API

RESTful 文档, 部分 MQTT API 没有的操作会在此

`http://example.com:8080/api`

2 数据格式

2.1 由节点下发至平台的二进制帧

以下简称为上行 (uplink)

下表中, 开始位为最低位. (也就是说, 开始位位于最后) (见例子)

`tables` 中有着协议的图形化解释, 可以用 Excel 打开.

2.1.1 ping 位置报送

开始 bit	结束 bit	字段含义	长度 byte	类型	范围	备注
0	7	消息类型	1	-	0x00~0xFF	在此固定为 0x8a
8	23	士兵编号	2	uint16	0~65535	-
24	55	纬度	4	-	-	见经纬度含义
56	87	经度	4	-	-	见经纬度含义
88	96	背甲击中次数	1	uint8	0~255	-
96	103	头盔击中次数	1	uint8	0~255	-
104	111	剩余子弹数量	1	uint8	0~255	-
112	119	冗余位	1	-	-	见冗余位含义

2.1.1.1 用例

`{冗余位}(00) {剩余弹药}(80) {头盔击中次数}(01) {背甲击中次数}(01) {经度}(43 53 12 F6) {纬度}(10 EC 1B 0`

```
00 80 01 01 43 53 12 F6 10 EC 1B 0E 00 7B 8A
```

```
{
    " 冗余位":0,
    " 剩余弹药":128,
    " 头盔击中次数":1,
    " 背甲击中次数":1,
    " 经度":172.3548549,
    " 纬度":43.326926,
    " 士兵编号":123,
```

```
    " 消息类型":"0x8a"
}
```

2.1.2 hit 击中

开始 bit	结束 bit	字段含义	长度 byte	类型	范围	备注
0	7	消息类型	1	-	0x00~0xFF	在此固定为 0x5b
8	23	士兵编号 (被击中者)	2	uint16	0~65535	-
24	39	士兵编号 (开枪者)	2	uint16	0~65535	-
40	40	头是否被击中	-	bit	0/1	-
41	41	-	-	bit	0/1	保留位
42	42	左手是否被击中	-	bit	0/1	-
43	43	右手是否被击中	-	bit	0/1	-
44	44	左脚是否被击中	-	bit	0/1	-
45	45	右脚是否被击中	-	bit	0/1	-
46	46	前甲是否被击中	-	bit	0/1	-
47	47	后甲是否被击中	-	bit	0/1	-

2.1.2.1 用例

{击中部位}(0b01000000 = 0x40) {开枪者士兵编号}(01 C8) {被击中士兵编号}(00 78) {消息类型}(0x5b)

```
40 01 C8 00 78 5B
```

```
{
    " 开枪士兵编号":456,
    " 被击中士兵编号":123,
    " 头":false,
    " 左手":false,
    " 右手":false,
    " 左脚":false,
    " 右脚":false,
    " 前甲":true,
    " 后甲":false,
    " 消息类型":"0x5b"
}
```

2.2 由平台向节点下发的二进制帧

以下简称为下行 (downlink)

借助 LoraWAN 的 fport 来区分语义

2.2.1 演习

fport 为 3. data 为 0 时开始演习, 为 1 时停止演习.

- 开始演习

```
{
    "fPort": 3,
    "data": "AA==" //base64 encoded 0x00
}
```

- 结束演习

```
{
    "fPort": 3,
    "data": "AQ==" //base64 encoded 0x01
}
```

2.2.2 重填子弹

fport 为 5. data 为装填子弹数目.

```
{
    "fPort": 5,
    "data": "Hg==" //base64 encoded 0x1E = 30
}
```

3 杂项

3.1 经纬度含义

根据 NMEA 0183 规范

- 纬度 ddmm.mmmm
- 经度 dddmm.mmmm

前两个 byte 为 小数点前 (ddmm/dddmm), 后两个 byte 为小数点后 (mmmm)

N, S, E, W 由冗余位中的两位确定. 默认为 N/E (假设在中国境内使用)

3.1.1 例子

4332.6926

- 原始的四个 byte 为 0x10, 0xec, 0x1b, 0x0e (10EC1B0E)
- 取前两个 byte 以 uint16 解析, 除以 100 得到 43.32
- 取后两个 byte 以 uint16 解析, 除以 1000000 得到 0.006926
- 最终经纬度为两者之和: 43.32+0.006926=43.326926
- 转换为度分秒 (DMS) 表示为 43° 19’ 36.9336’’

DMS 转换参考下列 Javascript 代码

```
function toDegreesMinutesAndSeconds(coordinate) {
    var absolute = Math.abs(coordinate);
    var degrees = Math.floor(absolute);
    var minutesNotTruncated = (absolute - degrees) * 60;
    var minutes = Math.floor(minutesNotTruncated);
    var seconds = Math.floor((minutesNotTruncated - minutes) * 60);

    return degrees + " " + minutes + " " + seconds;
}
```

3.2 冗余位含义

N, S, E, W 由冗余位中的两位确定.

目前没有用处, 值为 0x00.