蓝牙终端非标AoA广播API

版本	日期	备注
1.2.0	20220419	对该版本之前的所有版本进行整理，形成此文档
1.2.1	20220421	增加发射频率及对修正错误
1.2.2	20240815	增加7.125K激活信息-0x0F

1. 本协议文档（PDF版本时）可能不是最新版本；
2. 本协议文档描述，蓝牙终端非标AoA广播API；
3. 开发者若需要硬件对接请联系销售人员，https://www.imyfit.com ；
4. 若协议文档有更新，不再另行通知，强烈建议开发者到 https://imyfit.gitee.io 获取在线最新协议；
5. 若有疑问请到ISSUE区提出 https://gitee.com/imyfit/imyfit-issue 愿我们的付出对您的开发事半功倍。

1.概述

本协议主要描述4.x的蓝牙协议实现AoA定位，终端使用特定的蓝牙广播巧妙实现定位。

2.终端广播协议

任意4.x的终端设备使用如下的常规广播即可实现AoA定位。

字段名称	说明
报头(0x02)	1 octet，报文类型为不可连接广播
长度(0x25)	这个长度是指在 PDU 中的数据除去报头和长度之外的有效净荷数据长度
终端MAC地址	6 octets
长度(0x1E)	1 octet，制造商特定数据长度（三段式-长度）
类型(0xFF)	1 octet，制造商特定数据类型（三段式-类型）
Company ID(0x0D00)	2 octets，制造商ID，固定不变
Packet ID(0x04)	1 octet，包ID，固定不变
用户自定义数据	6 octets，后面两个字节为CRC16校验码，高字节发送在前低字节在后面 具体协议参照下文《4.用户自定义数据》
DF Field	20 octets，direction finding相关，固定内容： 0x2F,0x61,0xAC,0xCC,0x27,0x45,0x67,0xF7,0xDB,0x34, 0xC4,0x03,0x8E,0x5C,0x0B,0xAA,0x97,0x30,0x56,0xE6

3.终端配置协议

任意手机/网关发送如下的常规广播即可对终端参数进行配置。

字段顺序	说明
1	参数命令 bit[0:3]： 固定000 bit[4:5]： 告诉基站此时信标端当前的版本号; 00：方案 3 信标（原来信标方案） 01：方案 1 信标（公司提供 SDK 现在方案） 10：方案 2 信标（采用标准 SDK 协议栈） 11：保留 bit6： 告诉基站此时信标端是否开启接收窗口; 0：不开窗 1：开窗 bit7: 请求基站下行白化数据还是非白化数据; 0: 非白化（公司提供 SDK 的方案） 1：白化 （采用标准 SDK 协议栈）
2	bit[0:2]： 表示信道 000/2401信道 001/2402信道 010/2426信道 011/2480信道 100/2481信道 bit[3]： 最高位是否开启接收模式 1 表示一上电开启了接收模式 0 一上电没有开启 bit[4:7]： 接收模式，数值范围 0-9 对应信标发射功率 0000/0dBm 1000/ 3dBm 0100/4dBm 1100/-40dBm 0010/-20dBm 1010/-16dBm 0110/-12dBm 1110/-8dBm 0001/-4dBm 1001/-30dBm**
3	bit[0:2]： 信标设备类型 000/TI 001/Nordic Semiconductor，其他数字保留 bit[3]： 报警状态值 0/没有报警 1/报警 bit[4:7]：电池电量单位百分比，范围 0-10
4	bit[0:6]：发射频率（见发射频率表）
5	CRC高八位
6	CRC低八位

发射频率表

bit6	bit5	bit4 - bit0
1	0	表示Hz数，即[1,2,3……,31]等Hz
1	1	表示10*Hz数，即[10,20,30……310]等Hz值
0	0	表示周期秒数，即[1,2,3……,31]等秒数
0	1	表示10*秒数，即[10,20,30……310]等秒数

发射频率表【实例】

序号	频率Hz	周期s	bit6	bit5	bit4	bit3	bit2	bit1	bit0
1	300	0.003333	1	1	1	1	1	1	0
2	100	0.01	1	1	0	1	0	1	0
3	50	0.02	1	1	0	0	1	0	1
4	30	0.033333	1	0	1	1	1	1	0
5	20	0.05	1	0	1	0	1	0	0
6	10	0.01	1	0	0	1	0	1	0
7	9	0.111111	1	0	0	1	0	0	1
8	8	0.125	1	0	0	1	0	0	0
9	7	0.142857	1	0	0	0	1	1	1
10	6	0.166667	1	0	0	0	1	1	0
11	5	0.2	1	0	0	0	1	0	1
12	4	0.25	1	0	0	0	1	0	0
13	3	0.333333	1	0	0	0	0	1	1
14	2	0.5	1	0	0	0	0	1	0
15	1	1	1	0	0	0	0	0	1
16	0.5	2	0	0	0	0	0	1	0
17	0.2	5	0	0	0	0	1	0	1
18	0.1	10	0	0	0	1	0	1	0
19	0.05	20	0	0	1	0	1	0	0
20	0.02	50	0	1	0	0	1	0	1
21	0.01	100	0	1	0	1	0	1	0

4.用户自定义数据

用户自定义数据，分两类：默认数据类型（核芯物联已经规定成文，不可修改的数据）、其它数据类型（可根据需要添加）。实则数据域仅3个字节可使用。

4.1默认数据类型-0x00/0x08

1.？？数据0x00

2.加速度传感数据0x08

加速度传感数据x,y,z三轴分别占用8bit，有符号位（分辨率8bit）

字段顺序	说明
1	bit[0:3]参数命令0x08, bit[4:7]保留
2	传感数据 x轴
3	传感数据 y轴
4	传感数据 z轴
5	CRC高八位
6	CRC低八位

4.2其它数据类型-0x09~0x0F

其它数据为开发的，可根据需要修改，下表为其它数据类型总览。后文仅讨论3个字节的自定义数据。

字段顺序	说明
1	数据类型：0x09~0x0F，bit[4:7]保留
2	自定义数据（1octet）
3	自定义数据（1octet）
4	自定义数据（1octet）
5	CRC高八位（1octet）
6	CRC低八位（1octet）

1.设备状态-0x09

字段顺序	说明
数据类型	0x09即十进制09（数据0为该字段值）
数据1	bit[0]： 断带状态 0断带 1正常 bit[1]： 跌倒报警 1-报警；0-正常 bit[2]： 充电器插入状态，1-插入；0-未插入 bit[3]： 充电状态，1-充电中；0-未充电/充满 bit[4]： SOS状态，1-报警；0-正常 bit[5]： 佩戴 1佩戴，0未佩戴 bit[6]： 动静状态，1-动；0-静 bit[7]： 运动模式，1-开启；0-关闭
数据2	软件版本
数据3	电压值为（读数*6.6/255*）*V*。例如：读出*10*进制*159*则电压 为*(159/255)*6.6=4.1V **新版本修改为电量百分比，系统兼容方案为低于100则认为是电量百分比**

2.心率与血压-0x0A

字段顺序	说明
数据类型	0x0A即十进制10
数据1	心率（BPM）（十进制） 200：以下是正常心率 252：测量异常 250：未佩戴 251：心率模块异常 255：标签不支持心率模块 0：未完成测量
数据2	收缩压（十进制） 255：标签不支持心率模块 0：未完成测量
数据3	舒张压（十进制） 255：标签不支持心率模块 0：未完成测量

3.环境温度与血氧-0x0B

环境温度与血压

字段顺序	说明
数据类型	0x0B即十进制11
数据1	血氧（十进制） 255：标签不支持心率模块 0：未完成测量
数据2	未定义（保留0）
数据3	未定义（保留0）

4.体表温度与计步-0x0C

字段顺序	说明
数据类型	0x0C即十进制12
数据1	体表温度(值+200)/10; 例如156 表示为35.6摄氏度 0xAA
数据2	计步低字节
数据3	计步高字节

体温参考算法如下

```
z人体温度,y体表温度,x环境温度(均为浮点型数值)
y = (0xAA + 200)/10;
x = 25;
z=0.0337*y*y-0.545*y+1.7088*x-0.0519*x*y+17.626
```

5.运动数据-0x0D

字段顺序	说明
数据类型	0x0D即十进制13
数据1	热量低字节
数据2	热量高字节
数据3	睡眠状态 0x00：清醒 0x01：浅度睡眠 0X02：深度睡眠 0xFF：未检测

6.设备标示-0x0E

字段顺序	说明
数据类型	0x0E即十进制14
数据1	设备标识高字节，例如X3W，则为0x08
数据2	设备标识低字节，例如X3W，则为0x26
数据3	保留

7.125K激活信息-0x0F

字段顺序	说明
数据类型	0x0F即十进制15
数据1	rssi
数据2	基站ID
数据3	文本信息

5.CRC校验运算参考

```
/** CRC字节值表－高位 */
const unsigned char gClient_auchCRCHi[256] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40
} ;
```

```
/** CRC字节值表－低位 */
const unsigned char gClient_auchCRCLo[256] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06,
0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD,
0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A,
0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC, 0x14, 0xD4,
0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3,
0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,
0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29,
0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED,
0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
```

```
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60,
0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67,
0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68,
0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E,
0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71,
0x70, 0xB0, 0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92,
0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B,
0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B,
0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42,
0x43, 0x83, 0x41, 0x81, 0x80, 0x40 } ;
```

```
/**
 * @brief CRC16校验方法
 *
 * @param Len 数据长度
 * @param pBuf 数据指针
 * @return uint16_t 返回的校验值
 */
static uint16_t AoA_SendCrc16(uint16_t Len, uint8_t *pBuf)
{
    uint8_t uchCRCHi; /* 高CRC字节初始化 */
    uint8_t uchCRCLo; /* 低CRC字节初始化 */
    uint8_t uIndex;   /* CRC循环中的索引 */
    uint16_t CrcValue;

    uchCRCHi = 0xFF;
    uchCRCLo = 0xFF;
    while (Len--)     /* 传输消息缓冲区 */
    {
        uIndex = uchCRCHi ^ (*pBuf++); /* 计算CRC */
        uchCRCHi = uchCRCLo ^ gClient_auchCRCHi[uIndex];
        uchCRCLo = gClient_auchCRCLo[uIndex];
    }
    CrcValue = uchCRCHi;
    CrcValue <<= 8;
    CrcValue |= uchCRCLo;
    return CrcValue;
}
```

6.附录-广播示例

1.广播加速度传感器数据示例

```
0x02  //固定字节
0x25  //固定字节
0x01,0x02,0x03,0x04,0x05,0x06 //终端MAC地址，固定6个字节
0x1E  //固定字节
0xFF  //固定字节
0x0D  //固定字节
0x00  //固定字节
0x04  //固定字节
0x08  //加速度命令字
```

```
0x01  //加速度传感器x轴数据（注意这里读取加速度传感器原始x轴数据，分辨率8位）
0x01  //加速度传感器y轴数据（注意这里读取加速度传感器原始y轴数据，分辨率8位）
0x3E  //加速度传感器z轴数据（注意这里读取加速度传感器原始z轴数据，分辨率8位）
0xB7  //CRC16bit 校验码高字节 校验码前面15个字节运算的结果； 也就是从MAC地址开始计算到用户数据截止；
0xE6  //CRC16bit 校验码低字节；
0x2F,0x61,0xAC,0xCC,0x27,0x45,0x67,0xF7,0xDB,0x34,0xC4,0x03,0x8E,0x5C,0x0B,0xAA,0x97,0x30,0x56,0xE6//DF Field
```

2.广播用户数据示例

```
0x02  //固定字节
0x25  //固定字节
0x01,0x02,0x03,0x04,0x05,0x06 //MAC地址 固定6个字节
0x1E  //固定字节
0xFF  //固定字节
0x0D  //固定字节
0x00  //固定字节
0x04  //固定字节
0x09  //用户命令字 范围0x09-0x0F；
0x02  //用户自定义数据（注意这里0x02只作为示例，用户可自定义其他数据）
0x03  //用户自定义数据（注意这里0x03只作为示例，用户可自定义其他数据）
0x04  //用户自定义数据（注意这里0x04只作为示例，用户可自定义其他数据）
0xC7  //CRC16bit 校验码高字节 校验码前面15个字节运算的结果； 也就是从MAC地址开始计算到用户数据截止；
0x69  //CRC16bit 校验码低字节；
0x2F,0x61,0xAC,0xCC,0x27,0x45,0x67,0xF7,0xDB,0x34,0xC4,0x03,0x8E,0x5C,0x0B,0xAA,0x97,0x30,0x56,0xE6//DF Field
```

<<返回