**固件功能要求：**

1广播间隔默认200ms，最好可以配置100ms 、200ms、300ms、500ms、1s等间隔。

2 依次循环广播 0x00, 0x080,x09,0x0A,0x0B,0x0C,0x0D,0x0E 包上报用户数据，没有用到的数据包不上报。

3 默认广播信道 37 信道，可以通过配置切换 37，38，39 信道。平时广播为不可连接单通道广播，可以通过设置进行切换，切换成可连接广播进行OTA。

4 默认发射功率 0db（支持功率可配置-8dBm 到+4dBm）。

5手环佩戴才检测心率，不佩戴时不检测。心率初始上报250，上报数据规则见心率包说明，心率、血压和血氧逻辑：需要设置成间隔测量，间隔时间可设置，间隔支持1min、2min、3min、5min、10min、20min、30min等时间，心率和血氧测量过需要自动测量血氧，如果翻页至心率界面，则测量心率，然后测量血压、血氧，然后之前定时统计时间需要清0；如果翻页至血氧界面，则测量过血氧，定时统计时间直接清0。

6手环佩戴才检测体温，不佩戴时不体温。体温需要间隔测量，间隔时间可设置，间隔支持1min、2min、3min、5min、10min、20min、30min等时间，如果翻页至体温测量界面，直接测量体温，然后体温定时统计时间清0。

7计步和睡眠时间需实时统计，睡眠支持实时查看当日睡眠时间数据，睡眠和计步需要第二天0点数据清0。

**广播发送格式：**

|  |  |
| --- | --- |
| 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 |
| 发送MAC地址 | 6 octets  |
| 长度(0x1E) | 1 octet，数据类型«Manufacturer Specific Data»占用的数据长度 |
| 数据类型(0xFF) | 1 octet,Manufacturer Specific Data |
| Company ID(0x0D00) | 2 octets  |
| Packet ID(0x04) | 1 octet  |
| User data | 6 octets，注意后面两个字节为CRC16校验码，占用User data最后两个字节，高字节发送在前低字节在后面。具体协议请参照下面用户自定义数据协议 |
| DF field | BLE 37通道 发送内容为:0x2F,0x61,0xAC,0xCC,0x27,0x45,0x67,0xF7,0xDB,0x34,0xC4,0x03,0x8E,0x5C,0x0B,0xAA,0x97,0x30,0x56,0xE6 |
| BLE 38通道 发送内容为:0x6F,0xD3,0x10,0x0F,0x38,0x72,0x2D,0xA8,0x5E,0xC2,0x58,0x99,0x4F,0x8A,0xCE,0xEE,0xB7,0x69,0x88,0x07 |
| BLE 39通道 发送内容为:0x50,0xBD,0x84,0xB1,0x32,0x9F,0x14,0x9D,0xDD,0x6F,0xD3,0x10,0x0F,0x38,0x72,0x2D,0xA8,0x5E,0xC2,0x58 |

##### 信标端上行数据格式

User data 具体格式为：命令字(1Byte)+数据内容(3Bytes)+CRC16(2Bytes)

其中命令字 0x00-----》配置信息。0x08-------》上报加速度信息。命令字节范围0x09-0x0f代表其他用户数据，具体内容如下所示。

发射信道列表bit占用

|  |  |
| --- | --- |
| 000b ----->0 | 2401MHZ |
| 001b ----->1 | 2402MHZ |
| 010b ----->2 | 2426MHZ |
| 011b ----->3 | 2480MHZ |
| 100b ----->4 | 2481MHZ |

 表(1)

发射频率规则列表



 表(2)

|  |
| --- |
| Beacon类型 Nordic发射功率表 |
| 实际发射功率 | 对应数值大小 |
| 0 dBm | 0 |
| 3 dBm | 1 |
| 4 dBm | 2 |
| -40 dBm | 3 |
| -20 dBm  | 4 |
| -16 dBm  | 5 |
| -12 dBm | 6 |
| -8 dBm | 7 |
| -4 dBm | 8 |
| -30 dBm | 9 |

表(3)

|  |
| --- |
| Beacon类型 TI发射功率表 |
| 实际发射功率 | 对应数值大小 |
| 0 dBm | 0 |
| 1 dBm | 1 |
| 2 dBm | 2 |
| 3 dBm | 3 |
| 4 dBm  | 4 |
| 5 dBm  | 5 |
| -3 dBm | 6 |
| -6 dBm | 7 |
| -9 dBm | 8 |
| -12 dBm | 9 |

表示(4)

|  |
| --- |
| bit[0:3]：具体实现命令，具体含义如下所示 |
| bit[4-5]：告诉基站此时信标端当前的版本号; 00：方案3信标（原来信标方案提供SDK，默认是每隔4秒开启5ms的接收窗口同时支持私有频点，**注意在CCS显示为方式0，采用这种方案信标端需要上报该值**）01：方案1信标（提供SDK现在的上下行对齐方案，默认是每隔1秒请求基站下行数据同时支持私有频点，**注意在CCS显示为方式1，采用这种方案信标端需要上报该值**）10：方案2信标（采用标准官网提供的SDK，只支持37 38 39三个信道，不支持私有频点，**注意在CCS显示为方式2，采用这种方案信标端需要上报该值**）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（提供SDK的方案3和方案1，或者说通过2.4g私有协议完成）1：白化 （采用标准官网提供的SDK） |

 表(5)

配置信息协议 (命令 0x00) 注意CCS会通过该命令进行解析并显示出来如下截图1所示

|  |  |
| --- | --- |
| 1 | 参数命令bit[0:3]---->0x00, Bit[4:7]如表(5)所示 |
| 2 | bit[0:2]:发射信道(如上面列表1所示对应实际工作为五个信道)。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，0一上电没有开启接收模式：bit[4:7]:数值范围0-9 对应信标发射功率如表(3)、表(4)所示。 |
| 3 | bit[0:2]:信标设备类型000:TI,001:Nordic，其他数字保留。bit[3]:SOS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，规则如表(2)所示。bit[7]:保留 |
| 5 | CRC高八位 |
| 6 | CRC低八位 |



 截图1

加速度传感数据协议 注意 x,y,z三轴数据分别占用8bit，有符号位(分辨率8bit)，加速度传感器测量范围需要设置±2g。

|  |  |
| --- | --- |
| 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 |

**用户自定义数据协议（User data）**

|  |  |
| --- | --- |
| 1 | 参数命令---->(0x09---->0x0F)，bit[4:7]保留。（1octet） |
| 2 | 用户自定义数据（1octet） |
| 3 | 用户自定义数据（1octet） |
| 4 | 用户自定义数据（1octet） |
| 5 | CRC高八位（1octet） |
| 6 | CRC低八位（1octet） |

命令字0x09[设备状态]

|  |  |
| --- | --- |
| 命令字 | 0x09即十进制09（数据0为该字段值） |
| 数据1 | 第0-1位： 保留，默认值为0 （高位）第2位： 充电器插入状态 1插入；0未插入第3位： 充电状态 1充电中 0未充电/充满第4位： SOS状态 1 报警 0正常第5位： 佩戴状态 1佩戴 0脱腕 第6位： 动静状态 1动 0静第7位： 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/255\*6.6） V 举例：读出10进制159则电压为159/255\*6.6=4.1V |

命令字0x0A

|  |  |
| --- | --- |
| 命令字 | 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 | 255：标签不支持心率模块。0未完成测量，收缩压（十进制） |
| 数据3 | 255：标签不支持心率模块。0未完成测量，舒张压（十进制） |

命令字0x0B

|  |  |
| --- | --- |
| 命令字 | 0x0B即十进制11 |
| 数据1 | 血氧 |
| 数据2 | 环境温度低字节 0xBB |
| 数据3 | 环境温度高字节 0xCC |

体温参考算法：

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 = 0xCCBB \* 0.005;

z=0.0337\*y\*y-0.545\*y+1.7088\*x-0.0519\*x\*y+17.626

命令字0x0C

|  |  |
| --- | --- |
| 命令字 | 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 |

计步计算

比如低字节是0x56，高字节是0x78，那总计步数就是0x7856，再转成10进制就可以

命令字0x0D[运动数据] 睡眠状态 （0x00:清醒；0X01:浅度睡眠；0X02:深度睡眠；0XFF:未检测）

|  |  |
| --- | --- |
| 命令字 | 0x0D即十进制13 |
| 数据1 | 睡眠状态 （0x00:清醒；0X01:浅度睡眠；0X02:深度睡眠；0XFF:未检测） |
| 数据2 | 浅睡眠时间（统计小时和分钟，以10min为单位进行上传，比如5h30min，上报数据为：（5\*60+30）/10=33，上报33） |
| 数据3 | 深睡眠时间（统计小时和分钟，以10min为单位进行上传，比如5h30min，上报数据为：（5\*60+30）/10=33，上报33） |

命令字0x0E[设备标识]

|  |  |
| --- | --- |
| 命令字 | 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 |  |

**CRC16bit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 } ;

//CRC16

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;

}

**具体发送示例：**

 1:BLE 广播具体发送加速度传感器数据内容示例

 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

 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

 2:BLE 广播具体发送用户数据内容示例

 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