```
package com.hlk.hlkradartool.util

/**
 * 本类作为 2450 主动上报数据的实际数据的解析
 * 注：本类对于原始数据处理是按照字符串的解析方式，如果接收到的原始数据为字节数据，需将字节数组转为字符串
 */
class DataAnalysisDemo {
    private val notTargetDataStr = "0000000000000000"                    // 代表不检测某个目标区域或不存在这个区域，或者这个区域内没有目标信息
    private val singleTargetDataLength = 16                              // 每个目标的数据长度
    private var currentDataIndex = 8                                     // 记录当前数据解析解析到哪个数据了，从 8 开始是因为要除去帧头 4 个字节，即 8 个字符

    /**
     * 解析 2450 一直上报的区域检测信息
     * @param strData 上报的完整的原始数据
     * @return 包含有效的目标的实际数据的 map
     */
    fun analysisAreaCoordinate(strData:String):MutableMap<Int,TagInfoBean>{
        return strData.let{
            val strArray = mutableListOf<String>()
            // 循环三次，截取出三段目标数据
            for (i in 0 until 3){
                strArray.add(strData.substring(currentDataIndex,currentDataIndex + singleTargetDataLength))
                currentDataIndex += singleTargetDataLength
            }
            val map = mutableMapOf<Int,TagInfoBean>()
            for (i in strArray.indices) {
                if (strArray[i] != notTargetDataStr) {
                    // 原始数据中算出来的真实 x 坐标
                    val areaCoordinateXReal = calculateXValue(strArray[i])
                    // 原始数据中算出来的真实 y 坐标
                    val areaCoordinateYReal = calculateYValue(strArray[i])
                    // 计算实际的距离和角度
                    val areaCoordinateReal = calculateDistance(
                        areaCoordinateXReal,
                        areaCoordinateYReal
                    )
                    val areaAngleReal = calculateAngle(
                        areaCoordinateXReal,
                        areaCoordinateYReal
                    )
                    // areaCoordinateReal 实际距离，因为计算出来的距离单位是 mm，但实际需要显示的是 m，所以需要/1000
                    val bean = TagInfoBean(areaCoordinateXReal,areaCoordinateYReal,areaCoordinateReal / 1000.0,areaAngleReal)
                    map[i + 1] = bean
                }
            }
            map
```

```
        }
    }

    /**
     * 以下方法用于解析模块上报的区域检测数据，str 传递的原始数据格式为 aabb（以下数据说明均以 aabb 为例），即两个字节
     * 判断条件中比如 str.substring(2, 4).toInt(16) >= 128，是用于判断一个字节坐标数据中的第二位字节，即 bb 是否大于 128，即是否大于 0x80
     *  一个坐标数据两个字节，用字符串表示为 aabb，由于是小端序列，按照国人习惯的阅读方式，从左到右表示从小到大，在计算过程中第一和第二个字节需要互换位置，表现为 bbaa
     **/

    // 解析原始数据中某个区域的实际的 x 值
    private fun calculateXValue(str: String): Double {
        return if (str.substring(2, 4).toInt(16) >= 128) {
            ((str.substring(0, 2).toInt(16) + str.substring(2, 4).toInt(16) * 256) - 32768).toDouble()
        } else {
            (0 - (str.substring(0, 2).toInt(16) + str.substring(2, 4).toInt(16) * 256)).toDouble()
        }
    }

    // 解析原始数据中某个区域的实际的 y 值
    private fun calculateYValue(str: String): Double {
        return if (str.substring(6, 8).toInt(16) >= 128) {
            ((str.substring(4, 6).toInt(16) + str.substring(6, 8).toInt(16) * 256) - 32768).toDouble()
        } else {
            0 - (str.substring(4, 6).toInt(16) + str.substring(6, 8).toInt(16) * 256).toDouble()
        }
    }

    // 计算两个坐标点之间的实际距离
    private fun calculateDistance(x: Double, y: Double): Double {
        return Math.sqrt(Math.pow(x - 0.0, 2.0) + Math.pow(y - 0.0, 2.0));
    }

    // 计算两个坐标点之间的实际角度
    private fun calculateAngle(x: Double, y: Double): Double {
        // 计算与纵向下半轴的夹角（以度为单位）
        val angle = Math.atan2(y, x)
        // 因为这种计算方式是按照极坐标与 x 正轴之间的夹角算的，而实际需求是要极坐标与 y 正轴的之间的夹角算，所以 90° 减去与 x 正轴之间的夹角，剩下的夹角就是与 y 正轴的之间的夹角
        // 至于用 0 减去角度是因为视图和实际是旋转 180° 的，即视图的-x 为实际的 x 视图的 x 为实际的-x
        return 0-(90.0-Math.toDegrees(angle))
    }


    /**
     * 目标信息类
```

```
     * @param areaCoordinateXReal  实际 x 坐标
     * @param areaCoordinateYReal  实际 y 坐标
     * @param areaCoordinateReal   实际距离
     * @param areaAngleReal        实际角度
     */
    data class TagInfoBean(val areaCoordinateXReal:Double,val areaCoordinateYReal:Double,val
areaCoordinateReal:Double,val areaAngleReal:Double)
}
```