Jar 包调用说明

在 Demo 的程序首页中，可以选择 socket 通讯，跟串口通讯。根据设备自身的协议进行选择。设备的默认波特率是 9600

串口号选择，对应新平板的是 1，2，3，其中 1，2 是 RS232 串口，3 是 458。

返回函数没有的则代表不需要查看返回

RFID

根据需要选择串口号跟波特率。使用时需要先将标签放置到读取区域

1.1 初始化：

```
Util.ip  IP 地址 如 192.168.1.100
Util.port 端口  如：80
Util.serialPortIndex 串口号 1，2，3 对应平板的 3 个串口
Util.baud 波特率 如 9600 115200
DataBus dataBus;
if ("socket".equals(Util.MODE)) {
    dataBus = DataBusFactory.newSocketDataBus(Util.ip, Util.port);
} else {
    dataBus = DataBusFactory.newSerialDataBus(Util.serialPortIndex, Util.baud);
}
//接收的数据，data 为接收的数据，根据需求进行数据处理，注意在此处最好只解析一种的，
如果多种解析，由于返回的协议数据长度不一样可能会导致异常
dataBus.setReciveDataListener(new ReciveData() {
    @Override
    public String getReciveData(byte[] data) {
        return null;
    }
});

RFID rfid = new RFID(dataBus,null);
```

1.2 读取标签号

```
rfid.readSingleEpc(new SingleEpcListener() {
    @Override
    public void onVal(String val) {
        //val 是读取到的标签号
```

```
    }

    @Override
    public void onFail(Exception e) {

Toast.makeText(getApplicationContext(),e.toString(),Toast.LENGTH_SHORT).show();
    }
```

1.3 标签数据写入

```
其中 data 是需要写入的字符串
rfid.writeData(data, new RFIDWriteListener() {
    @Override
    public void onResult(boolean isSuccess) {
        Toast.makeText(ActivityRFID.this.getApplicationContext(), isSuccess + "",
Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onFail(Exception e) {

Toast.makeText(getApplicationContext(),e.toString(),Toast.LENGTH_SHORT).show();
    }
});
```

1.4 读取标签数据

```
rfid.readData(new RFIDReadListener() {
    @Override
    public void onResult(String str) {
        //str 是读取的标签数据
    }

    @Override
    public void onFail(Exception e) {

Toast.makeText(getApplicationContext(),e.toString(),Toast.LENGTH_SHORT).show();
    }
});
```

串口类

初始化

```
DataBus dataBus;
if ("socket".equals(Util.MODE)) {
    dataBus = DataBusFactory.newSocketDataBus(Util.ip, Util.port);
} else {
    dataBus = DataBusFactory.newSerialDataBus(Util.serialPortIndex, Util.baud);
}
//接收的数据，data 为接收的数据，根据需求进行数据处理，注意在此处最好只解析一种的，如果多种解析，由于返回的协议数据长度不一样可能会导致异常
dataBus.setReciveDataListener(new ReciveData() {
    @Override
    public String getReciveData(byte[] data) {
        return null;
    }
});
GenericConnector genericConnector = new GenericConnector(dataBus, new ConnectResultListener() {
    @Override
    public void onConnectResult(boolean isSuccess) {
        System.out.println(isSuccess);
    }
});
//如果需要查看原始返回报文的，可以调用此方法，返回 byte 字节。如果需要转 16 进制字符串可以使用方法 DataTools.formatByteArray(data).进行字符串输出查看
dataBus.getReceiveData()
```

多合一

多合一地址查询

地址查询：sendAllInOneGetAddress（ConnectorListener listener)
ConnectorListener 回调，回调 onSuccess 返回的是是否发送成功，根据返回值，需要用对应的函数进行解析。后文的回调都是一致的
返回地址获取：getAllInOneGetAddress。
例子：

```
genericConnector.sendAllInOneGetAddress(new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("多合一从机地址:
```

```
"+genericConnector.getAllInOneGetAddress());
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

多合一查询 PM2.5

PM2.5 查询： sendAllInOnePM25（int address, ConnectorListener listener）
Address 设备地址，listener 回调
返回数据读取函数：`getAllInOnePm25`
用法如上例子

多合一查询人体

人体查询： sendAllInOneBody（int address, ConnectorListener listener）
Address 设备地址，listener 回调
返回数据读取函数：`getAllInOneBody`
用法如上例子

多合一查询空气质量

空气质量查询： sendAllInOneAirQuality（int address, ConnectorListener listener）
Address 设备地址，listener 回调
返回数据读取函数：`getAllInOneAirQuality`
用法如上例子

多合一查询温湿度

温湿度查询： sendAllInOneTempHum（int address, ConnectorListener listener）
Address 设备地址，listener 回调
返回数据读取函数：温度：`getAllInOneTemp`，湿度：`getAllInOneHum`
用法如上例子

多合一查询大气压

大气压查询： sendAllInOnePressure（int address, ConnectorListener listener）
Address 设备地址，listener 回调
返回数据读取函数：getAllInOnePressure
用法如上例子

百叶箱

数据查询：sendLouverBoxValue（ConnectorListener listener）
listener 回调
返回数据读取函数：温度：getTemperature，湿度：getHumidity
用法如上例子

二氧化碳变送器 485

二氧化碳变送器地址设置 fe 广播方式

地址查询： sendSet485Co2Adress（int address, ConnectorListener listener）
Address 新设备地址，listener 回调

二氧化碳变送器地址查询

查询： sendGet485Co2ValueByAddress（int address, ConnectorListener listener）
Address 设备地址，listener 回调
返回数据读取函数：get485Co2Adress
用法如上例子

二氧化碳变送器数据查询

数据查询： sendGet485Co2Value（ConnectorListener listener）
listener 回调
返回数据读取函数：get485Co2Value
用法如上例子

北斗模块

北斗模块版本号查询

函数： sendGPSVersion（int address, ConnectorListener listener）
Address 设备地址，listener 回调
返回数据读取函数：getGPSVersion
用法如上例子

北斗地址查询

地址查询： sendGPSAddress（ConnectorListener listener）
listener 回调
返回数据读取函数：getGPSAddress
用法如上例子

北斗模块波特率查询

函数： sendGPSBps（int address, ConnectorListener listener）
Address 设备地址，listener 回调
返回数据读取函数：getGPSBps
用法如上例子

北斗模块是否奇偶校验查询

函数： sendGPSOdd（int address, ConnectorListener listener）
Address 设备地址，listener 回调
返回数据读取函数：getGPSOdd。00 则无奇偶校验，01 有
用法如上例子

北斗模块定位数据查询

函数： sendGetGPSData（int address, ConnectorListener listener）
Address 设备地址，listener 回调
返回数据读取函数：getGPSData。返回得是 GPSDATA 数据

```java
public class GPSDATA {
    private String longitude;//精度 ddmm.mmmmm计算要转为度: 36度 + 40.46260分。40.46260/60=0.67438度,所以为36.67438度
    private String latitude;//维度 ddmm.mmmmm
    private String RMC;//协议类型头
    private String utcTime;//时间 时分秒秒
    private String positionState;//定位状态A 有效 V 无效
    private String longitudeDirection;//精度方向  E-东经，W-西经
    private String latitudeDirection;;//维度方向  N-北纬，S-南纬
    private String speed;//对地速度
    private String direction; //对地航向
    private String date;//日期年月日
    private String wifiState;//天线状态 OK 代表天线正常 OK；OP 代表 OPEN；OR 代表天线短路 SHORT
    private String originData;//未转换的源数据字符串
```

用法如上例子

北斗模块地址设置查询

函数： sendSetGPSAddress（int address, ConnectorListener listener）
Address 新设备地址，listener 回调
返回数据读取函数：getSetGPSAddress
用法如上例子

北斗模块波特率设置查询

函数： sendSetGPSBps（int address, int bate, ConnectorListener listener）
Address 设备地址，bate 波特率 0-7 -->0 1 2 3 4 5 6 7 8 9 对应 1200 - 115200 默认 3，listener 回调
返回数据读取函数：getSetGPSBps
用法如上例子

水浸变送器

水浸变送器获取数据

函数： sendWaterImmersion（int address, ConnectorListener listener）
Address 设备地址，listener 回调
返回数据读取函数：getSetWaterImmersion
用法如上例子

超声波

超声波获取数据

函数： sendUltrasonicValue（int address, int type, ConnectorListener listener）

Address 设备地址，type 数据种类，0 读取处理值,1 读取实时值,2 读取温度值 ledListener 回调
返回数据读取函数：getUltrasonicValue
用法如上例子

超声波设置地址

函数： sendSetUltrasonicAddress（int address, ConnectorListener listener）
Address 新设备地址，listener 回调
返回数据读取函数：getSetUltrasonicAddress
用法如上例子

Led 综合显示屏 4 路

发送显示屏文字

函数： LedScreenCmd（String txt,,ConnectorListener listener）
txt 要发送的文字数字，listener 回调
返回数据读取函数：getSetUltrasonicAddress
用法如上例子

485 调速电机

调速电机设置波特率

函数： sendEleBate（int address,int type,ConnectorListener listener）
Address 设备地址，**type** 填入波特率如 9600，listener 回调
返回数据读取函数函数：getEleAddress
用法如上例子

调速电机获取地址

函数： sendEleGetAddress（ConnectorListener listener）
listener 回调
返回数据读取函数函数：getEleAddress
用法如上例子

调速电机设置地址

函数： sendEleSetAddress（int address,ConnectorListener listener）
Address 新设备地址，listener 回调
返回数据读取函数函数：getEleAddress
用法如上例子

调速电机查询频率

函数： sendEleGetFrequency（int address,int type,ConnectorListener listener）
Address 新设备地址，type 1=电机 1，2=电机 2 ,listener 回调
返回数据读取函数函数：getEleGetFrequency
用法如上例子

调速电机设置频率

函数： sendEleSetFrequency（int address,int type,int bate,ConnectorListener listener）
Address 新设备地址，type 1=电机 1，2=电机 2 , bate 1000HZ≤PWM 频率≤10000HZ（十进制）,listener 回调
返回数据读取函数函数：getEleGetFrequency
用法如上例子

调速电机读取电机方向

函数： sendEleGetDirection（int address,int type,ConnectorListener listener）
Address 设备地址，type 1=电机 1，2=电机 2 ,listener 回调
返回数据读取函数函数：getEleGetDirection 获取方向 0 正 1 负
用法如上例子

调速电机设置方向

函数： sendEleSetDirection（int address,int type,int direction,ConnectorListener listener）
Address 设备地址，type 1=电机 1，2=电机 2 , direcion 0 正向 1 反向,listener 回调
返回数据读取函数函数：getEleSetDirection 获取方向 0 正 1 负
用法如上例子

调速电机读取速度

函数： sendEleGetSpeed（int address,int type,ConnectorListener listener）
Address 设备地址，type 1=电机 1，2=电机 2 ,listener 回调
返回数据读取函数函数：getEleGetSpeed
用法如上例子

调速电机设置速度

函数： sendEleSetSpeed（int address,int type,int speed,ConnectorListener listener）
Address 设备地址，type 1=电机 1，2=电机 2 , speed 速度 0-100，listener 回调
返回数据读取函数函数：getEleSetSpeed
用法如上例子

调速电机刹车

函数： sendEleStop（int address,int type,ConnectorListener listener）
Address 设备地址，type 1=电机 1，2=电机 2，listener 回调
返回数据读取函数函数：getEleSetSpeed
用法如上例子

联动控制器 404D

联动控制器 out 控制

函数： sendLinkControlOut（int address,int count,boolean isOpen,ConnectorListener listener）
address 查询地址（可用广播 fe），count 0-3 共四路，isOpen true 打开 false 关闭，listener 回调
返回数据读取函数函数：
用法如上例子

联动控制器 in 光耦读取

函数： sendLinkReadIn（int address,int count,ConnectorListener listener）

address 查询地址，count 0-3 共四路，listener 回调
返回数据读取函数函数：getSearchLinkOutIn
用法如上例子

联动控制器 out 读取

函数： sendSearchLinkOut（int address, int count,ConnectorListener listener）
address 查询地址，count 0-3 共四路，listener 回调
返回数据读取函数函数：getSearchLinkOutIn
用法如上例子

联动控制器 闪开闪关（间隔一秒）

函数： sendLinkFlashOnOff（int address,Boolean,isOpen,ConnectorListener listener）
address 设备地址，isOpen true 闪开，false 闪关，listener 回调
返回数据读取函数函数：
用法如上例子

联动控制器 全开全关

函数： sendLinkAllOnOff（int address,Boolean,isOpen,ConnectorListener listener）
address 设备地址 ,isOpen true 开，false 关，listener 回调
返回数据读取函数函数：
用法如上例子

Zigbee

Zigbee 单双联继电器控制

函数： ZigbeeControl(int serialNum,byte choose,ConnectorListener listener)
high 高位在前，low 低位在后(如 0x1122)，choose 类型，第一路开，第一路关，第二路开，第二路，枚举如:Const.SecondClose

```
 //第一路开，关
    public static final byte FirstOpen = 0x01;
    public static final byte FirstClose = 0x02;
    //第二路，开关
```

```
    public static final byte SecondOpen = 0x10;
    public static final byte SecondClose = 0x20;
```

listener 回调

返回数据读取函数函数：

用法如上例子

Zigbee 解析传感器数据

Zigbee 传感数据解析在另外一个类 Zigbee3 中，用法：

```
 ZigBee3 zigBee3 = new ZigBee3(data);
 if(zigBee3.sensorType()==ZigBeeSensorType.TEM_HUM.getCode())
 {
    Toast.makeText(ActivityGenericConnector.this,"===温湿度
== "+zigBee3.getVal0(),Toast.LENGTH_SHORT).show();
 }

switch (zigBee3.sensorType())
{
   case 0x01:
      Toast.makeText(ActivityGenericConnector.this,"===温湿度
== "+zigBee3.getVal0(),Toast.LENGTH_SHORT).show();
      break;
   case 0x02:
      Toast.makeText(ActivityGenericConnector.this,"===人体
== "+zigBee3.getVal0(),Toast.LENGTH_SHORT).show();
      break;
   case 0x03:
      Toast.makeText(ActivityGenericConnector.this,"===火焰
== "+zigBee3.getVal0(),Toast.LENGTH_SHORT).show();
      break;
}
```

```
public enum ZigBeeSensorType {
    TEM_HUM( name: "温湿度传感器",  code: 0x01),
    WEIGHT( name: "重力传感器",  code: 0x02),
    GYRO( name: "陀螺仪",  code: 0x03),
    PERSON( name: "人体传感器",  code: 0x11),
    LIGHT( name: "光照传感器",  code: 0x21),
    CO( name: "CO传感器(空气质量)",  code: 0x22),
    COM_GAS( name: "可燃气传感器",  code: 0x23),
    FIRE( name: "火焰传感器",  code: 0x24),
    ALCOHOL( name: "酒精",  code: 0x25),
    FOUR_ENTER( name: "四通道电流",  code: 0x30),;
    private final String name;
    private final int code;
```

RGB 灯带

初始化：

```
DataBus dataBus;
if ("socket".equals(Util.MODE)) {
    dataBus = DataBusFactory.newSocketDataBus(Util.ip, Util.port);
} else {
    dataBus = DataBusFactory.newSerialDataBus(Util.serialPortIndex, Util.baud);
}

//接收的数据，data 为接收的数据，根据需求进行数据处理，注意在此处最好只解析一种的，如果多种解析，由于返回的协议数据长度不一样可能会导致异常
dataBus.setReciveDataListener(new ReciveData() {
    @Override
    public String getReciveData(byte[] data) {
        return null;
    }
});

RgbLed rgbLed = new RGBLed(dataBus, new ConnectResultListener() {
            @Override
            public void onConnectResult(boolean isSuccess) {
                System.out.println(isSuccess);
            }
        });
```

RGB 灯带单路控制：

函数：controlRGBOne(int code,boolean isOpen,int address, ConnectorListener listener)
code 第几路有 1，2，3 路，isOpen true 打开 false 关闭，address 设备地址（默认 fe 广播查询）,listener 回调
返回数据读取函数函数：getRec
用法如上例子

RGB 灯带三色控制控制：

函数：controlRGB(int blue,int green,int red, int address, ConnectorListener listener)
Blue 蓝色 0-255，green 绿色 0-255，红色 0-255，address 设备地址（默认 fe 广播查询）,listener 回调
返回数据读取函数函数：getRec
用法如上例子

UWB 高精度定位

UWB 设备会主动上报数据，我们需要将四个位置到信号接收器的距离获取，再经过公式的转换获取到对应的坐标。
再数据接收的地方进行解析，最后获取 x,y 的坐标。坐标系可以只定义一次进行使用，后续每次获取到 r1,r2,r3,r4，都装载到方法 Trilateration.GetLocation，进行获取坐标即可。

```
dataBus.setReciveDataListener(new ReciveData() {
    @Override
    public String getReciveData(byte[] data) {
        UWB uwb=genericConnector.getUWBData(data);
//坐标系 位置如下
//A3        A2
//A0        A1
List<UWBData> anchorArray=new ArrayList<>();
//单位毫米的  r0,r1,r2,r3 的值。注意单位
int r1,r2,r3,r4;
r1= Integer.valueOf(uwb.getR1());
r2= Integer.valueOf(uwb.getR2());
r3= Integer.valueOf(uwb.getR3());
r4= Integer.valueOf(uwb.getR4());
//坐标系的x,y 轴. 单位米
double x =2.27;
double y =1.57;
int[] Range_deca = new int[]{r1*10,r2*10,r3*10,r4*10};
//四个坐标系
UWBData report=new UWBData();
```

```
//A0
UWBData uwbData = new UWBData();
uwbData.x=0.0;
uwbData.y=0.0;
uwbData.z=2.0;
anchorArray.add(uwbData);
//A1
UWBData uwbData1 = new UWBData();
uwbData1.x= x; //anchor2.x uint:m
uwbData1.y= 0.0; //anchor2.y uint:m
uwbData1.z=2.0;
anchorArray.add(uwbData1);
//A2
UWBData uwbData2 = new UWBData();
uwbData2.x= x; //anchor2.x uint:m
uwbData2.y= y; //anchor2.y uint:m
uwbData2.z=2.0;
anchorArray.add(uwbData2);
//A3
UWBData uwbData3 = new UWBData();
uwbData3.x= 0.0; //anchor2.x uint:m
uwbData3.y= y; //anchor2.y uint:m
uwbData3.z=2.0;
anchorArray.add(uwbData3);
Trilateration.anchorArray = anchorArray;
Trilateration.distanceArray = Range_deca;
int result = Trilateration.GetLocation(anchorArray,Range_deca);
//坐标的（x,y）中的x = Trilateration.best_solution.x   ,y =
Trilateration.best_solution.y
Log.e("test","  x  "+Trilateration.best_solution.x+"  y
"+Trilateration.best_solution.y+"   z  "+Trilateration.best_solution.z);

        return null;
    }
});
```

IOT 采集器（包含 tcp 模式，与串口的 rtu 模式）

IOT 采集器地址查询-tcp

函数： sendTCPIOTAddress（ConnectorListener listener）

listener 回调
返回数据读取函数函数：getTCPIOTAddress
用法如上例子

IOT 采集器地址修改-tcp

函数： sendSetIOTaddress（int oldAddress,int newaddress,ConnectorListener listener）
oldAddress 旧地址， address 新设备地址，listener 回调
返回数据读取函数函数：getTCPIOTSetAddress 新地址
用法如上例子

IOT 采集器读取输入寄存器（模拟电流）-tcp

函数： sendTCPgetIOTVirtData（int address,int code,ConnectorListener listener）
Address 设备地址，code 0 2 4 ,代表 A0 A2 A4 ,listener 回调
返回数据读取函数函数：getTCPIOTVirtData 对应的值
用法如上例子

IOT 采集器读取读取离散寄存器（DI）-tcp

函数： sendTCPReadDI（int address,ConnectorListener listener）
Address 设备地址 ,listener 回调
返回数据读取函数函数：getTCPReadDI 此处获取的是 DI 数据类型，对应 D1-D8，可根据需要的数据进行,如果要取 DI1 则 getDI0

```
package com.nle.mylibrary.protocolEntity.modbus;

public class DI {
    private int DI0;
    private int DI1;
    private int DI2;
    private int DI3;
    private int DI4;
    private int DI5;
    private int DI6;
    private int DI7;
```

用法如上例子

IOT 采集器写入单个线圈寄存器（DO）-tcp

函数： sendTCPSetDoVlue (int address,int code,boolean isOpen,ConnectorListener listener)
Address 设备地址，code 1-8 对应 8 个 DO, isOpen true 打开，false 关闭 ledListener 回调
返回数据读取函数函数：getTCPIOTVirtData 对应的值
用法如上例子

IOT 采集器地址查询-rtu

函数： sendRtuIotAddress（ConnectorListener listener）
listener 回调
返回数据读取函数函数：getRtuIOTAddress
用法如上例子

IOT 采集器地址修改-rtu

函数： sendRtuSetIotAddress（int newaddress,ConnectorListener listener）
Address 新设备地址，listener 回调
返回数据读取函数函数：getRtuSetIOTAddress 新地址
用法如上例子

IOT 采集器读取输入寄存器（模拟电流）-rtu

函数： sendRtugetIOTVirtData（int address,int code,ConnectorListener listener）
Address 设备地址，code 0 2 4 ,代表 A0 A2 A4 ,listener 回调
返回数据读取函数函数：getRtuIOTVirtData 对应的值
用法如上例子

IOT 采集器写入单个线圈寄存器（DO）-rtu

函数： sendRtuWriteData (int address,int number,boolean isOpen,ConnectorListener listener)
Address 设备地址，number 1-8 对应 8 个 DO, isOpen true 打开，false 关闭 ledListener 回调
返回数据读取函数函数：getTCPIOTVirtData 对应的值
用法如上例子

IOT 采集器读取读取离散寄存器（DI）-rtu

函数： sendRtuReadDI（int address,ConnectorListener listener）
Address 设备地址 ,listener 回调
返回数据读取函数函数：getRtuReadDI 此处获取的是 DI 数据类型，对应 D1-D8，可根据需要的数据进行,如果要取 DI1 则 getDI0

```
package com.nle.mylibrary.protocolEntity.modbus;

public class DI {
    private int DI0;
    private int DI1;
    private int DI2;
    private int DI3;
    private int DI4;
    private int DI5;
    private int DI6;
    private int DI7;
```

用法如上例子