Jar 包调用说明

特别说明：

在程序中，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

在调用接口功能时，需要加上 try catch 抛出异常，使用效果如下

```
});
//二氧化碳变送器获取
findViewById(R.id.co2_485).setOnClickListener((view) -> {
        try {
            genericConnector.sendGet485Co2Value( address: 1,new ConnectorListener() {
                @Override
                public void onSuccess(boolean val) {
                    RealValue.setText("二氧化碳变送器485: " + genericConnector.get485Co2Value());
                }

                @Override
                public void onFail(Exception e) {

                }
            });
        } catch (Exception e) {
            e.printStackTrace();
        }
});
//设置二氧化碳变送器获取数据的地址
```

后续详细的接口调用，还会再次进行举例说明。

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;
//socket 模式
dataBus = DataBusFactory.newSocketDataBus(“192.168.0.101”, 80);
//串口模式
dataBus = DataBusFactory.newSerialDataBus(1, 9600);
//绑定
GenericConnector GenericConnector = new GenericConnector(dataBus, new
ConnectResultListener() {
    @Override
    public void onConnectResult(boolean isSuccess) {
        System.out.println(isSuccess);
    }
});
//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;
    }
});

//如果需要查看原始返回报文的，也可以调用此方法，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
```

多合一

多合一地址查询

地址查询：sendAllInOneGetAddress（GenericConnector genericConnector）
GenericConnector 回调，回调 onSuccess 返回的是是否发送成功，居提返回值，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。
使用实例：

```
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, GenericConnector genericConnector）
Address 设备地址，genericConnector 回调
返回数据读取函数：**getAllInOnePm25**
使用实例：

```
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) {

    }
});
```

多合一查询人体

人体查询： sendAllInOneBody（int address, GenericConnector genericConnector）
Address 设备地址，genericConnector 回调
返回数据读取函数：`getAllInOneBody`
使用实例：

```
GenericConnector.sendAllInOneBody(1,new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("多合一人体："+GenericConnector.getAllInOneBody());
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

多合一查询空气质量

空气质量查询： sendAllInOneAirQuality（int address, GenericConnector genericConnector）
Address 设备地址，genericConnector 回调
返回数据读取函数：`getAllInOneAirQuality`
使用实例：

```
GenericConnector.sendAllInOneAirQuality(1,new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("多合一空气质量：
"+GenericConnector.getAllInOneAirQuality());
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

多合一查询温湿度

温湿度查询： sendAllInOneTempHum（int address, GenericConnector genericConnector）

Address 设备地址，genericConnector 回调
返回数据读取函数：温度：getAllInOneTemp，湿度：getAllInOneHum
使用实例：

```
GenericConnector. sendAllInOneTempHum (1,new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("多合一温度："+GenericConnector.getAllInOneTemp ()+“湿度”
+ GenericConnector.getAllInOneHum ());
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

多合一查询大气压

大气压查询： sendAllInOnePressure（int address, GenericConnector genericConnector）
Address 设备地址，genericConnector 回调
返回数据读取函数：getAllInOnePressure
使用实例：

```
GenericConnector. sendAllInOnePressure (1,new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("多合一大气压："+GenericConnector. getAllInOnePressure ());
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

百叶箱

数据查询：sendLouverBoxValue（GenericConnector genericConnector）
genericConnector 回调
返回数据读取函数：温度：getTemperature，湿度：getHumidity
使用实例：

```
GenericConnector. sendLouverBoxValue (new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("百叶箱温度："+GenericConnector.getTemperature ()+“湿度”+
GenericConnector.getHumidity ());
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

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, GenericConnector genericConnector）
Address 新设备地址，genericConnector 回调

二氧化碳变送器数据查询

查询： sendGet485Co2Value（int address, GenericConnector genericConnector）
Address 设备地址，genericConnector 回调
返回数据读取函数：get485Co2Adress
使用实例：

```
GenericConnector. sendGet485Co2Value (1,new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("二氧化碳变送器数据："+GenericConnector. get485Co2Adress
());
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

北斗模块

北斗模块版本号查询

函数： sendGPSVersion（int address, GenericConnector genericConnector）
Address 设备地址，genericConnector 回调
返回数据读取函数：getGPSVersion
使用实例：

```
GenericConnector. sendGPSVersion (1,new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("北斗版本号："+GenericConnector. getGPSVersion ());
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

北斗地址查询

地址查询： sendGPSAddress（GenericConnector genericConnector）
genericConnector 回调
返回数据读取函数：getGPSAddress
使用实例：

```
GenericConnector. sendGPSAddress (new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("北斗地址："+GenericConnector. getGPSAddress ());
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

北斗模块波特率查询

函数： sendGPSBps（int address, GenericConnector genericConnector）
Address 设备地址，genericConnector 回调
返回数据读取函数：getGPSBps
使用实例：

```
GenericConnector. sendGPSBps (1,new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("北斗波特率："+GenericConnector. getGPSBps ());
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

北斗模块是否奇偶校验查询

函数： sendGPSOdd（int address, GenericConnector genericConnector）
Address 设备地址，genericConnector 回调
返回数据读取函数：getGPSOdd。00 则无奇偶校验，01 有
使用实例：

```
GenericConnector. sendGPSOdd (1,new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("北斗奇偶校验："+GenericConnector. getGPSOdd ());
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

北斗模块定位数据查询

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

使用实例：

```
GenericConnector. sendGetGPSData (1,new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("北斗经纬度："+GenericConnector. getLongitude
()+"  "+  GenericConnector. getLatitude());
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

北斗模块地址设置查询

函数：sendSetGPSAddress（int oldAddress, int address, GenericConnector genericConnector）

oldAddress 旧地址，Address 新地址，genericConnector 回调

返回数据读取函数：getSetGPSAddress

使用实例：

```
GenericConnector. sendSetGPSAddress (1,3,  new ConnectorListener() {
    @Override
```

```
        public void onSuccess(boolean val) {
            RealValue.setText("北斗新地址："+GenericConnector. getSetGPSAddress
());
        }

        @Override
        public void onFail(Exception e) {

        }
});
```

北斗模块波特率设置

函数： sendSetGPSBps（int address, int bate, GenericConnector genericConnector）
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,
genericConnector 回调
返回数据读取函数：getSetGPSBps
使用实例：

```
GenericConnector. sendSetGPSBps (1,3,new ConnectorListener() {
        @Override
        public void onSuccess(boolean val) {
            RealValue.setText("北斗新地址："+GenericConnector. getSetGPSBps());
        }

        @Override
        public void onFail(Exception e) {

        }
});
```

水浸变送器

水浸变送器获取数据

函数： sendWaterImmersion（int address, GenericConnector genericConnector）
Address 设备地址，genericConnector 回调
返回数据读取函数：getSetWaterImmersion
使用实例：

```
GenericConnector. sendWaterImmersion (1,new ConnectorListener() {
        @Override
        public void onSuccess(boolean val) {
```

```
        RealValue.setText("北斗新地址："+GenericConnector. getSetWaterImmersion
());
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

超声波

超声波获取数据

函数： sendUltrasonicValue（int address, int type, GenericConnector genericConnector）
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
genericConnector 回调
返回数据读取函数：getUltrasonicValue
使用实例：

```
GenericConnector. sendUltrasonicValue (1,1,new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("超声波数据："+GenericConnector. getUltrasonicValue());
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

超声波设置地址

函数： sendSetUltrasonicAddress（**int** address, **int** newAddress
, GenericConnector genericConnector）
address 旧地址 newAddress 新设备地址，genericConnector 回调
返回数据读取函数：getSetUltrasonicAddress
使用实例：

```
GenericConnector. sendSetUltrasonicAddress(1,3,new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("超声波地址："+GenericConnector.
getSetUltrasonicAddress());
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

Led 综合显示屏 4 路

发送显示屏文字

函数：LedScreenCmd（int address,int x,int y,int textsize,int textColor,String txt,GenericConnector genericConnector）

address 设备的地址
x 文字显示的 x 轴位置 0-128
y 文字显示的 y 轴位置 0-128
textsize 字体大小 12 14 16 才有效，其他的字体需要下载字库
textColor 字体的颜色 0 黑色（完全看不到） 1 红色
txt 要发送的文字数字，genericConnector 回调

使用实例：

```
GenericConnector. LedScreenCmd(1,32,32,14,1,"测试文字",new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {

    }

    @Override
    public void onFail(Exception e) {

    }
});
```

清楚显示屏内容

函数： ClearLedScreen（int address,GenericConnector genericConnector）
address 设备的地址
调用此方法将会清除屏幕上的所有内容
返回数据读取函数：无
使用实例：

```
GenericConnector. ClearLedScreen (1,new ConnectorListener() {

    @Override
    public void onSuccess(boolean val) {
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

485 调速电机

调速电机设置波特率

函数： sendEleBate（int address,int type,GenericConnector genericConnector）
Address 设备地址，type 填入波特率如 9600，genericConnector 回调
返回数据读取函数函数：
使用实例：

```
GenericConnector. sendEleBate (1,9600,new ConnectorListener() {

    @Override
    public void onSuccess(boolean val) {
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

调速电机获取地址

函数： sendEleGetAddress（GenericConnector genericConnector）
genericConnector 回调
返回数据读取函数函数：getEleAddress
使用实例：

```
GenericConnector. sendEleGetAddress (new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("485 调速电机地址："+GenericConnector. getEleAddress ());
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

调速电机设置地址

函数： sendEleSetAddress（int address,GenericConnector genericConnector）
Address 新设备地址，genericConnector 回调
返回数据读取函数函数：getEleAddress
使用实例：

```
GenericConnector. sendEleSetAddress (new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("485 调速电机地址："+GenericConnector. getEleAddress ());
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

调速电机查询频率

函数： sendEleGetFrequency（int address,int type,GenericConnector genericConnector）
Address 新设备地址，type 1=电机 1，2=电机 2 ,genericConnector 回调
返回数据读取函数函数：getEleGetFrequency
使用实例：

```
GenericConnector. sendEleGetFrequency (1,1,new ConnectorListener() {

    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("485 调速电机 1 的频率："+GenericConnector.

getEleGetFrequency());

    }

    @Override
    public void onFail(Exception e) {

    }
});
```

调速电机设置频率

函数： sendEleSetFrequency（int address,int type,int bate,GenericConnector genericConnector）
Address 新设备地址，type 1=电机 1，2=电机 2 , bate 1000HZ≤PWM 频率≤10000HZ（十进制）,genericConnector 回调
返回数据读取函数函数：getEleGetFrequency
使用实例：

```
GenericConnector. sendEleSetFrequency (1,1,1000,new ConnectorListener() {

    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("485 调速电机 1 的频率："+GenericConnector.

getEleGetFrequency());

    }

    @Override
    public void onFail(Exception e) {
```

```
    }
});
```

调速电机读取电机方向

函数： sendEleGetDirection（int address,int type,GenericConnector genericConnector）
Address 设备地址，type 1=电机 1，2=电机 2 ,genericConnector 回调
返回数据读取函数函数：getEleGetDirection 获取方向 0 正 1 负
使用实例：

```
GenericConnector. sendEleGetDirection (1,1,new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("485 调速电机 1 的方向："+GenericConnector.
getEleGetDirection ());
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

调速电机设置方向

函数： sendEleSetDirection（int address,int type,int direction,GenericConnector genericConnector）
Address 设备地址，type 1=电机 1，2=电机 2 , direcion 0 正向 1 反向,genericConnector 回调
返回数据读取函数函数：getEleSetDirection 获取方向 0 正 1 负
使用实例：

```
GenericConnector. sendEleSetDirection(1,1,0,new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("485 调速电机 1 的方向："+GenericConnector.
getEleSetDirection());
    }

    @Override
    public void onFail(Exception e) {
```

```
    }
});
```

调速电机读取速度

函数： sendEleGetSpeed（int address,int type,GenericConnector genericConnector）
Address 设备地址，type 1=电机 1，2=电机 2 ,genericConnector 回调
返回数据读取函数函数：getEleGetSpeed
使用实例：

```
GenericConnector. sendEleGetSpeed(1,1,new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("485 调速电机 1 的速度："+GenericConnector.
getEleGetSpeed());
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

调速电机设置速度

函数： sendEleSetSpeed（int address,int type,int speed,GenericConnector genericConnector）
Address 设备地址，type 1=电机 1，2=电机 2 ,speed 速度 0-100，genericConnector 回调
返回数据读取函数函数：getEleSetSpeed
使用实例：

```
GenericConnector. sendEleSetSpeed (1,1,100,new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("485 调速电机 1 的速度："+GenericConnector.
getEleSetSpeed ());
    }

    @Override
```

```
    public void onFail(Exception e) {

    }
});
```

调速电机刹车

函数： sendEleStop（int address,int type,GenericConnector genericConnector）
Address 设备地址，type 1=电机 1，2=电机 2，genericConnector 回调
返回数据读取函数函数：
使用实例：

```
GenericConnector. sendEleStop (1,1,new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {

    }

    @Override
    public void onFail(Exception e) {

    }
});
```

联动控制器 404D

联动控制器 out 控制

函数： sendLinkControlOut（int address,boolean isOpen,GenericConnector genericConnector）
address 设备地址 0-3 共四路，isOpen true 打开 false 关闭，genericConnector 回调
返回数据读取函数函数：
使用实例：

```
GenericConnector. sendLinkControlOut (1,true,new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {

    }

    @Override
```

```
    public void onFail(Exception e) {

    }
});
```

联动控制器 in 光耦读取

函数： sendLinkReadIn（int address,GenericConnector genericConnector）
address 设备地址 0-3 共四路，genericConnector 回调
返回数据读取函数函数：getSearchLinkOutIn
使用实例：

```
GenericConnector. sendLinkReadIn (1, new ConnectorListener() {

    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("联动控制器 in 光耦读取："+GenericConnector.
getSearchLinkOutIn());

    }

    @Override
    public void onFail(Exception e) {

    }
});
```

联动控制器 out 读取

函数： sendSearchLinkOut（int address,GenericConnector genericConnector）
address 设备地址 0-3 共四路，genericConnector 回调
返回数据读取函数函数：getSearchLinkOutIn
使用实例：

```
GenericConnector. sendSearchLinkOut (1, new ConnectorListener() {

    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("联动控制器 out 读取："+GenericConnector.
getSearchLinkOutIn ());

    }
```

```
    @Override
    public void onFail(Exception e) {

    }
});
```

联动控制器 闪开闪关（间隔一秒）

函数： sendLinkFlashOnOff（boolean isOpen,GenericConnector genericConnector）
isOpen true 闪开，false 闪关，genericConnector 回调
返回数据读取函数函数：
使用实例：

```
GenericConnector. sendLinkFlashOnOff (true, new ConnectorListener() {

    @Override
    public void onSuccess(boolean val) {
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

联动控制器 全开全关

函数： sendLinkAllOnOff（boolean isOpen,GenericConnector genericConnector）
isOpen true 开，false 关，genericConnector 回调
返回数据读取函数函数：
使用实例：

```
GenericConnector. sendLinkAllOnOff (true, new ConnectorListener() {

    @Override
    public void onSuccess(boolean val) {
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

Zigbee

Zigbee 单双联继电器控制

函数： ZigbeeControl(int serialNum,byte choose,GenericConnector genericConnector)
serialNum 设备的短地址，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 = 0x21;
public static final byte FirstClose = 0x22;
//第二路，开关
public static final byte SecondOpen = 0x11;
public static final byte SecondClose = 0x12;
```

genericConnector 回调
返回数据读取函数函数：
使用实例：

```
GenericConnector. ZigbeeControl(0xAA04, Const.FirstOpen, GenericConnector() {
    @Override
    public void onSuccess(boolean val) {
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

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：

```
dataBus.setReciveDataListener(new ReciveData() {
      @Override
      public String getReciveData(byte[] data) {
      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;
}
            return null;
     }
});
```

RGB 灯带

```
初始化
DataBus dataBus;
//socket 模式
dataBus = DataBusFactory.newSocketDataBus(“192.168.0.101”, 80);
//串口模式
dataBus = DataBusFactory.newSerialDataBus(1, 9600);
//绑定
RgbLed rgbLed = new RGBLed(dataBus, new ConnectResultListener() {
                @Override
                public void onConnectResult(boolean isSuccess) {
                    System.out.println(isSuccess);
                }
        });
//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;
```

```
    }
});
```

RGB 灯带单路控制：

函数：controlRGBOne(int code,int value, int address,GenericConnector genericConnector)
code 第几路有 0，1，2 路，value 输入 0-255 进行色值的设置，genericConnector 回调
address 设备地址
返回数据读取函数函数：
使用实例：

```
rgbLed. controlRGBOne (0,255,1, new ConnectorListener() {

    @Override
    public void onSuccess(boolean val) {
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

RGB 灯带三色控制控制：

函数：controlRGB(int blue,int green,int red,int address, GenericConnector genericConnector)
blue 蓝色 0-255，green 绿色 0-255，红色 0-255，genericConnector 回调
address 设备地址
返回数据读取函数函数：getRec
使用实例：

```
rgbLed. controlRGB (128,128,128,1, new ConnectorListener() {

    @Override
    public void onSuccess(boolean val) {
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

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（GenericConnector genericConnector）
genericConnector 回调
返回数据读取函数函数：getTCPIOTAddress
使用实例：

```
GenericConnector. sendTCPIOTAddress (new ConnectorListener() {

    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("iot 采集器地址："+GenericConnector. getTCPIOTAddress ());
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

IOT 采集器地址修改-tcp

函数： sendSetIOTaddress（int newaddress,GenericConnector genericConnector）
Address 新设备地址，genericConnector 回调
返回数据读取函数函数：getTCPIOTSetAddress 新地址
使用实例：

```
GenericConnector. sendSetIOTaddress (3,new ConnectorListener() {

    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("iot 采集器新地址："+GenericConnector. getTCPIOTSetAddress
());
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

Tcp 修改 iot 的地址后需要重启设备新地址才能生效

IOT 采集器读取输入寄存器（模拟电流）-tcp

函数： sendTCPgetIOTVirtData（int address,int code,GenericConnector genericConnector）
Address 设备地址，code 0 2 4 ,代表 A0 A2 A4 ,genericConnector 回调
返回数据读取函数函数：getTCPIOTVirtData 对应的值
使用实例：

```
GenericConnector. sendTCPgetIOTVirtData (3,0,new ConnectorListener() {

    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("iot 采集器 A0 值："+GenericConnector. getTCPIOTVirtData ());
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

IOT 采集器读取读取离散寄存器（DI）-tcp

函数： sendTCPReadDI（int address,GenericConnector genericConnector）
Address 设备地址 ,genericConnector 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;
```

使用实例：

```
GenericConnector. sendTCPReadDI (3,new ConnectorListener() {

    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("iot 采集器 di0 值："+GenericConnector. getTCPReadDI
().getDI0);
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

IOT 采集器写入单个线圈寄存器（DO）-tcp

函数： sendTCPSetDoVlue （int address,int code,boolean isOpen,GenericConnector genericConnector）
Address 设备地址, code 1-8 对应 8 个 DO, isOpen true 打开, false 关闭 genericConnector 回调
返回数据读取函数函数：getTCPIOTVirtData 对应的值
使用实例：

```
GenericConnector. sendTCPReadDI (3,1,true,new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {
        //第一个 do1 打开
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

IOT 采集器地址查询-rtu

函数： sendRtuIotAddress（GenericConnector genericConnector）
genericConnector 回调
返回数据读取函数函数：getRtuIOTAddress
使用实例：

```
GenericConnector. sendRtuIotAddress (new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("iot 采集器地址："+GenericConnector. getRtuIOTAddress
());
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

IOT 采集器地址修改-rtu

函数： sendRtuSetIotAddress（int newaddress,GenericConnector genericConnector）
Address 新设备地址，genericConnector 回调
返回数据读取函数函数：getRtuSetIOTAddress 新地址
使用实例：

```
GenericConnector. sendRtuSetIotAddress (3,new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("iot 采集器新地址："+GenericConnector. getRtuSetIOTAddress
());
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

IOT 采集器读取输入寄存器（模拟电流）-rtu

函数： sendRtugetIOTVirtData（int address,int code,GenericConnector genericConnector）
Address 设备地址，code 0 2 4 ,代表 A0 A2 A4 ,genericConnector 回调
返回数据读取函数函数：getRtuIOTVirtData 对应的值
使用实例：

```
GenericConnector. sendRtugetIOTVirtData (3,0,new ConnectorListener() {
    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("iot 采集器 A0 值："+GenericConnector. getRtuIOTVirtData ());
    }

    @Override
    public void onFail(Exception e) {

    }
});
```

IOT 采集器写入单个线圈寄存器（DO）-rtu

函数： sendRtuWriteData （int address,int number,boolean isOpen,GenericConnector genericConnector）
Address 设备地址，number 1-8 对应 8 个 DO，isOpen true 打开，false 关闭 genericConnector 回调
返回数据读取函数函数：
使用实例：

GenericConnector. sendRtuWriteData (3,1,true,new ConnectorListener() {

    @Override
    public void onSuccess(boolean val) {
        //第一个 do1 打开
    }

    @Override
    public void onFail(Exception e) {

    }
});

IOT 采集器读取读取离散寄存器（DI）-rtu

函数： sendRtuReadDI（int address,GenericConnector genericConnector）
Address 设备地址 ,genericConnector 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;
```

使用实例：

GenericConnector. sendRtuReadDI (3,new ConnectorListener() {

    @Override
    public void onSuccess(boolean val) {
        RealValue.setText("iot 采集器 di0 值："+GenericConnector. getRtuReadDI().getDI0);
    }

    @Override
    public void onFail(Exception e) {

```
        }
});
```