物联网通讯协议(iot-modbus)开发指南

编写人：腾飞开源

2022.12.01

目录

1. 介绍 3
2. 开源项目 3
3. 版本说明 3
4. 软件架构 4
5. 安装教程 4
6. 工程结构 5
7. 使用说明 5
8. 指令格式 6
9. 通讯指令 7
10. 调试说明 8
11. 点亮 Star 12
12. 技术交流 12

1.介绍

物联网通讯协议，基于 netty 框架，支持 COM（串口）和 TCP 协议，支持服务端和客户端两种模式，实现 Java 控制智能设备，同时支持设备组多台设备高并发通讯。采用工厂设计模式，代码采用继承和重写的方式实现高度封装，可作为 SDK 提供封装的接口，让具体的业务开发人员无需关心通讯协议的底层实现，直接调用接口即可使用。实现了心跳、背光灯、扫码、刷卡、指静脉、温湿度和门锁（支持多锁）、LCD 显示屏等指令控制、三色报警灯控制。代码注释丰富，包括上传和下发指令调用例子，非常容易上手。

2.开源项目

（1）gitee 地址：https://gitee.com/takeoff/iot-modbus

（2）github 地址：https://github.com/luorongxi/iot-modbus

3.版本说明

（1） V1.0.0 版本仅支持 TCP 服务端通讯模式；

（2）V2.0.0 版本支持 TCP 服务端和客户端两种模式，客户端模式还增加了心跳重连机制。

（3）V3.0.0 版本支持 COM（串口）和 TCP 协议，增加 logback 日志按文件大小和时间切割输出。

（4） V3.1.0 版本代码优化，抽取公共模块子工程。

（5）V3.2.0 版本 TCP 通讯增加支持 LCD 显示屏控制指令，支持批量控制 LCD 显示屏。

（6）V3.2.1 版本串口通讯增加支持 LCD 显示屏控制指令，支持批量控制 LCD 显示屏。

（7）V3.2.2 版本串口通讯接收指令数据拆包处理代码优化，网口通讯增加支持三色报警灯控制指令。

（8）V3.2.3 版本串口通讯增加支持三色报警灯控制指令，串口通讯接收指令数据拆包处理代码优化。

（9）V3.2.4 版本使用 netty 集成 Rxtx 对串口数据进行数据拆包处理，并且对指静脉指令进行优化。

（10）V3.2.5 版本客户端模式支持同时连接多个服务端下发和接收指令数据。

4.软件架构

基础架构采用 Spring Boot2.x + Netty4.X + Maven3.6.x，日志采用 logback。

5.安装教程

（1）系统 Windows7 以上；

（2）安装 Jdk1.8 以上；

（3）安装 Maven3.6 以上；

（4）将代码以 Maven 工程导入 Eclipse 或 Idea。

6.工程结构

- iot-modbus //物联网通讯父工程
- ├── doc //文档管理
- ├── iot-modbus-client //netty 通讯客户端
- ├── iot-modbus-common //公共模块子工程
- ├── iot-modbus-netty //netty 通讯子工程
- ├── iot-modbus-serialport //串口通讯子工程
- ├── iot-modbus-server //netty 通讯服务端
- ├── iot-modbus-test //使用样例子工程
- └── tools //通讯指令调试工具

7.使用说明

（1）配置文件查看 iot-modbus-test 子工程 resources 目录下的 application.yml 文件；

（2）启动文件查看 iot-modbus-test 子工程 App.java 文件；

（3）服务启动后，服务端端口默认为：8080，网口通讯端口默认为：4000，串口通讯默认串口为：COM1；

（4）通讯指令调试工具，TCP 通讯模式使用 tools 目录下的 NetAssist.exe，串口通讯模式使用 tools 目录下的 UartAssist.exe；

（5）通讯指令采用 Hex 编码（十六进制）；

（6）串口通讯依赖文件查看 doc/serialport 目录，Windows 环境下 rxtxParallel.dll 和 rxtxSerial.dll 文件拷贝到 JDK 安装的 bin 目录下，Linux 环境将 librxtxParallel.so 和 librxtxSerial.so 文件拷贝到 JDK 安装的 bin 目录下；

（7）postman 指令下发样例请查看 doc/postman/通讯指令下发.postman_collection.json 文件，包括门锁（单锁/多锁）、扫码、背光灯、LCD 显示屏、三色报警灯指令。

8.指令格式

（1）以心跳指令（7E 04 00 BE 01 00 00 74 77 7F）作为样例说明，下标从 0 开始；

（2）第 0 位为起始符，长度固定占 1 个字节，固定格式：7E；

（3）第 1、2 位为数据长度，计算方法是从命令符到数据位（即：从 3 位到指令长度-3 位），长度固定占 2 个字节，例如：04 00，表示长度为 4；

（4）第 3 位为指令符，长度固定占 1 个字节，例如：BE，表示心跳指令；

（5）第 4 位为设备号，长度固定占 1 个字节，例如：01，表示设备号为 1；

（6）第 5 位为层地址，长度固定占 1 个字节，例如：00，表示设备所有的层不执行；

（7）第 6 位为槽地址，长度固定占 1 个字节，例如：00，表示设备所有的槽不执行；

（8）指令长度-3 位到-2 位为校验位，采用 CRC16_MODBUS（长度,命令,地址,数据）校验，例如：74 77，详细查看：ModbusCrc16Utils.java 工具类；

（9）末位为结束符，长度固定占 1 个字节，固定格式：7F。

9.通讯指令

（1）心跳上传指令

iot-netty 作为服务端，通过心跳来维持通讯，启动服务端后，打开 NetAssist.exe 指令调试工具，先往服务端发送心跳指令；

硬件往服务端发送：7E 04 00 BE 01 00 00 74 77 7F ，为必要指令。

（2）背光灯上传指令

硬件往服务端发送：7E 05 00 88 01 00 00 00 AF E3 7F ；

（3）扫码指令下发

服务端往硬件下发：7E 05 00 08 01 00 00 01 6F FD 7F ；

第 7 位为数据位，长度固定占 1 个字节，例如：01，表示开开启扫码头。

（4）扫码指令上传

硬件往服务端发送：7E 24 00 8F 01 00 00 03 45 30 30 34 30 31 30 38 32 38 30 32 41 36 39 33 0D 02 00 00 01 02 13 73 02 00 00 01 02 13 73 9B 79 7F；

数据为：03 45 30 30 34 30 31 30 38 32 38 30 32 41 36 39 33 0D 02 00 00 01 02 13 73 02 00 00 01 02 13 73 为条码信息。

（5）刷卡指令上传

硬件往服务端发送：7E 08 00 84 01 00 00 86 14 AE 02 7C 53 7F ；

数据位：86 14 AE 02 为卡号信息。

（6）单开锁下发指令

服务端往硬件下发：7E 05 00 03 01 00 00 01 CA 3C 7F；

第 7 位为数据位，长度固定占 1 个字节，例如：01，表示开 1 号锁。

（7）多开锁下发指令

服务端往硬件下发：7E 08 00 03 FF FF FF 01 00 02 01 7F B0 7F；

FF FF FF 为指令做兼容填补位，后面 01 00 02 01 是数据位，其中：01 表示 1 号锁，00 表示上锁；02 表示 2 号锁，01 表示开锁。

8. 锁状态上传指令

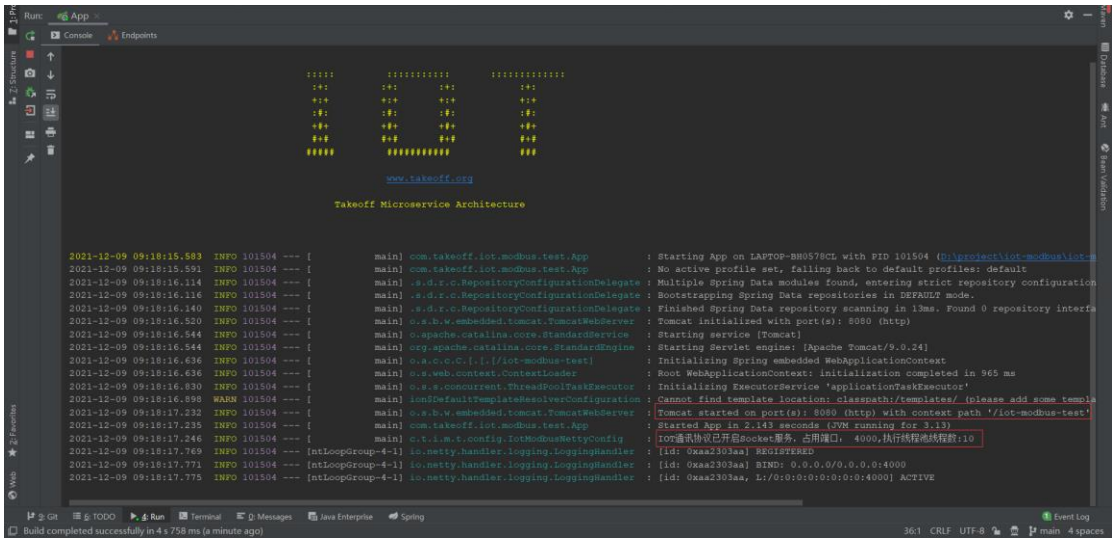
硬件往服务端发送：7E 0D 00 83 01 00 00 FF FF FF 01 00 05 02 00 01 EE 99 7F ；

FF FF FF 为指令做兼容填补位，后面 01 00 05 02 00 01 是数据位，其中：01 表示 1 号锁，00 表示上锁，05 表示传感器状态码；02 表示 2 号锁，00 表示上锁，01 表示传感器状态码。

9. 指静脉和温湿度指令（不作说明，详细查看代码）。

10. 调试说明

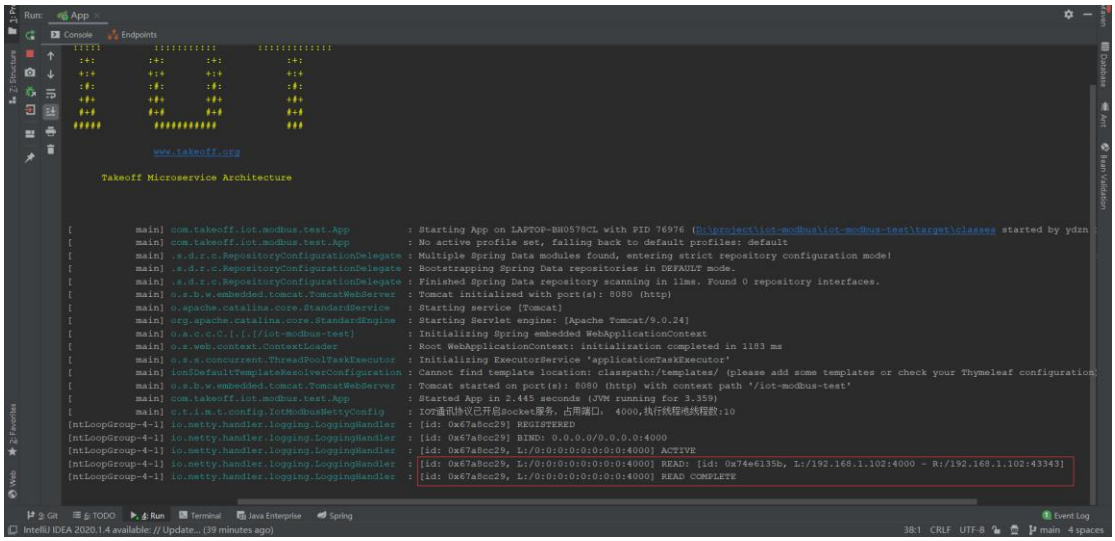
（1）找到 iot-server-api 子工程 App.java 文件启动服务端，如下图所示：



说明：项目启动成功后，控制台日志输出服务端的端口为：8080；项目服务名为：iot-modbus-test；服务端开启 socket 通讯端口为：4000。

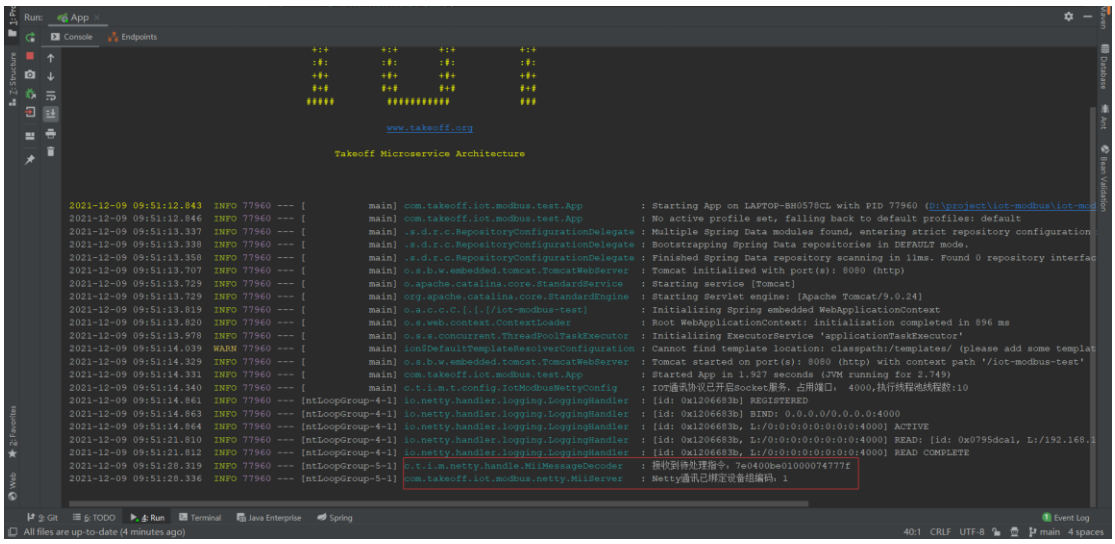
（2）将工程 tools 目录下通讯指令调试工具 NetAssist 拷贝到 Windows 桌面，双击打开，并配置参数，如下图所示：

说明：协议类型选择 TCP Client（调试工具作为模拟硬件通讯的客服端）；远程主机地址输入本地电脑的 IP 地址；远程主机端口输入服务端端口 4000；接收和发送的编码选择 HEX；最后点击连接按钮进行连接，连接成功后服务端控制台日志输出如下图所示：



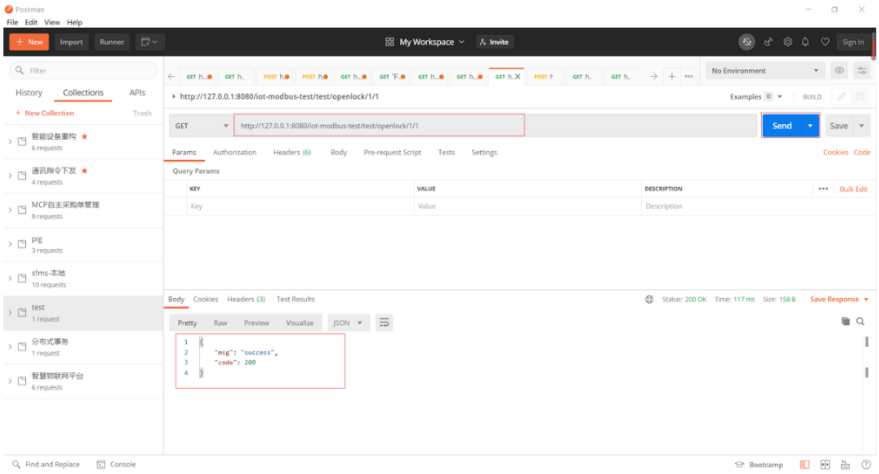
（3）客户端往服务端上传心跳指令，如下图所示：

说明：拷贝心跳指令到通讯指令调试工具 NetAssist 的数据发送窗口粘贴进来，然后点击发送按钮；此时服务端将接收到心跳指令，如下图所示：

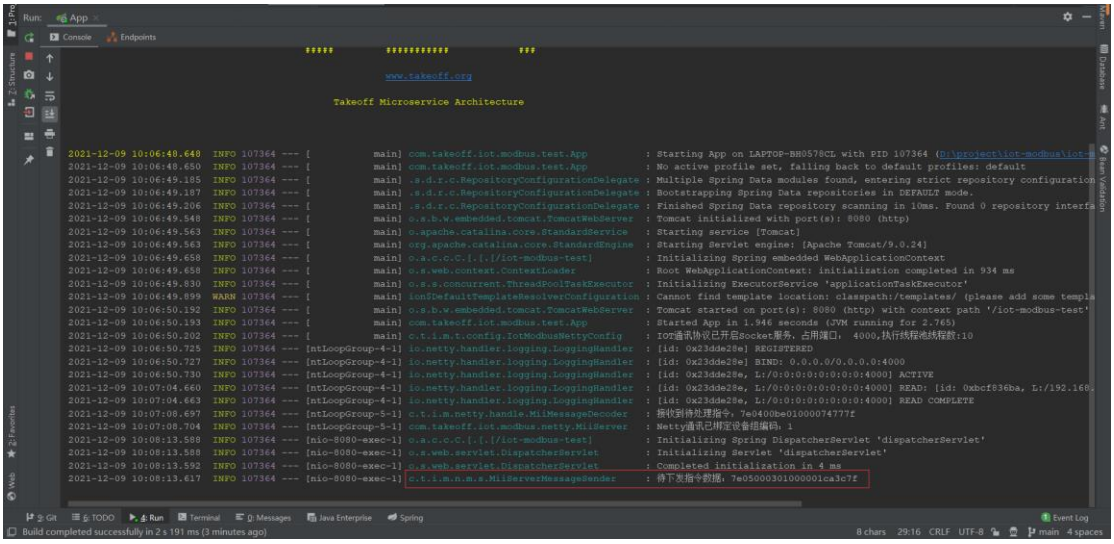


说明：客服端与服务端的通讯连接需要通过客户端定时往服务端发送心跳指令来维持，在生产环境中发送频率一般可设置为每 5 秒一次，如果通讯连接断开则客服端与服务端无法通讯。注意：在调试的过程中，如果通讯指令调试工具 NetAssist 与服务端通讯连接断开，则要手动点击 NetAssist 的连接按钮，重新往服务端发送一条心跳的指令。

（4）使用 Postman 请求服务端往客户端下发控制单锁指令，如下图所示：



说明：在 Postman 输入服务端发送控制单锁指令接口，填入请求地址和参数：http://127.0.0.1:8080/iot-modbus-test/test/openlock/1/1，服务端控制台日志输出如下图所示：



客服端指令调试工具 NetAssist 接收到控制单锁指令，如下图所示：

（5）其他上传和下发的指令不作详细介绍，感兴趣的同学可以参考通讯指令和工程目录 doc/postman 目录下的请求样例。

11. 点亮 Star

创作不易，别忘了点亮 Star，你们的支持，是我源源不断的动力。

12. 技术交流

（1）微信公众号

（2）QQ 技术交流群