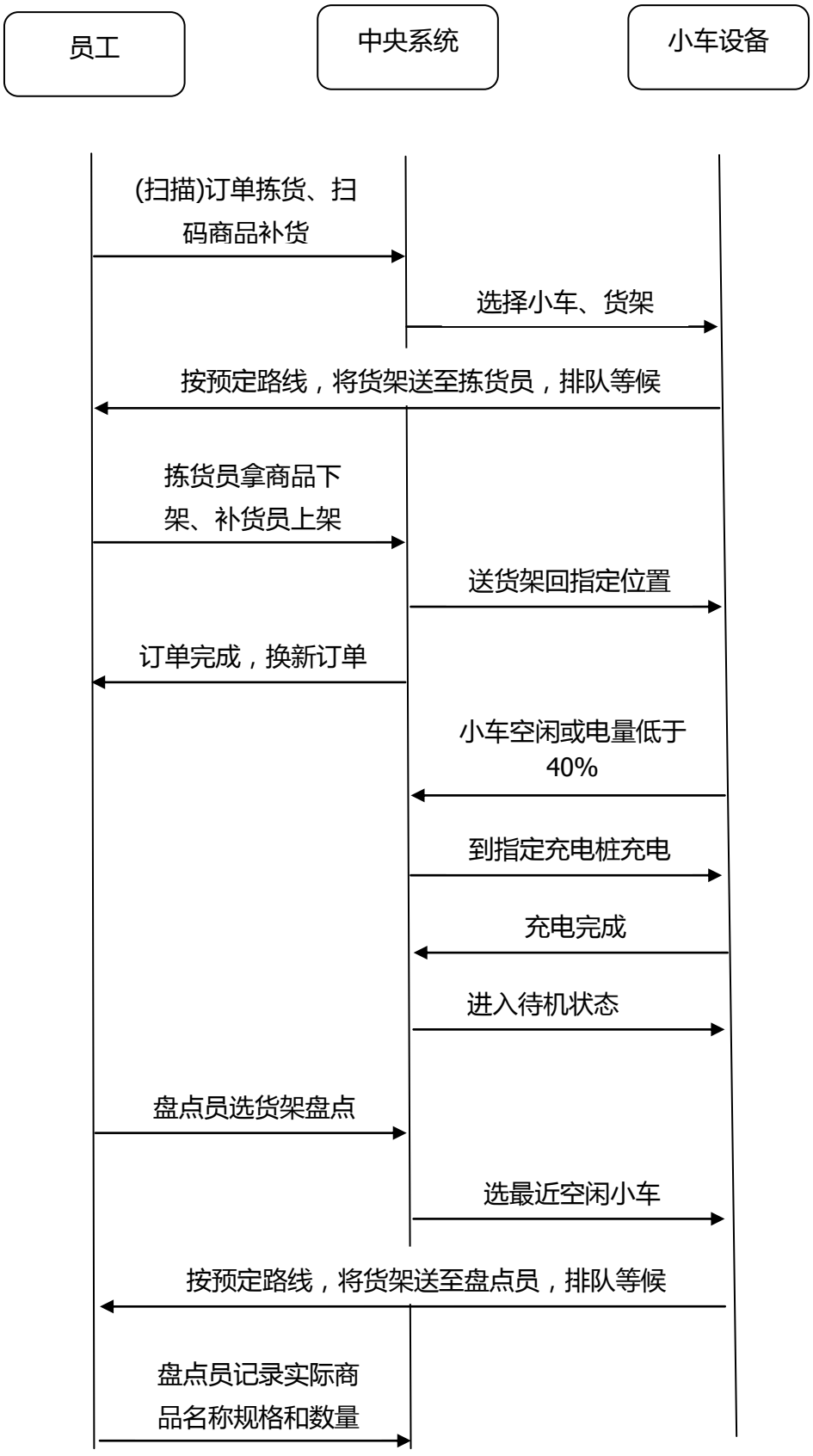
目录

## 功能概要设计

1. 产品上架
    a) 扫码仓库库位条码/直接手动输入库位编号（库区-通道-货架-层号-格号）
    b) 再依次扫码产品/直接录入产品 ID(按箱/组合上架产品暂不考虑)
2. 选择订单开始打包商品，绑定订单和盛放商品的容器，简称订单容器
3. 获取订单中的产品
    a) 选择小车设备（可能当前有多个设备可用）和目标货架（可能有多个货架上有对应商品）
    b) 分配设备取货（设备自行制定行走路线，给出相对坐标值，只需要向减少间距方向移动即可）
    c) 拣货员将商品从货架下，扫码/直接出库产品，然后拿至订单容器
    d) 设备归位货架（同 b，都属于在两点间通过设备移动货架）
4. 循环 2 中步骤，直至订单完成，换新订单
5. 安排设备充电

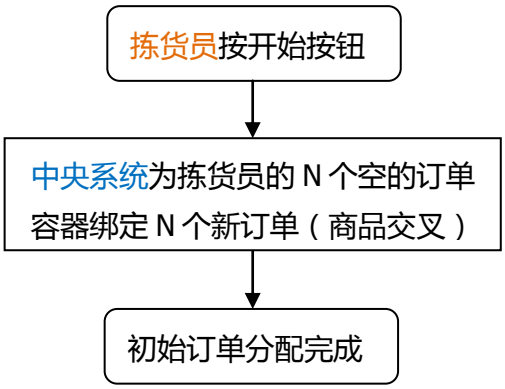
## 业务顺序图


员工
中央系统
小车设备
(扫描)订单拣货、扫码商品补货
选择小车、货架
按预定路线，将货架送至拣货员，排队等候
拣货员拿商品下架、补货员上架
送货架回指定位置
订单完成，换新订单
小车空闲或电量低于40%
到指定充电桩充电
充电完成
进入待机状态
盘点员选货架盘点
选最近空闲小车
按预定路线，将货架送至盘点员，排队等候
盘点员记录实际商品名称规格和数量

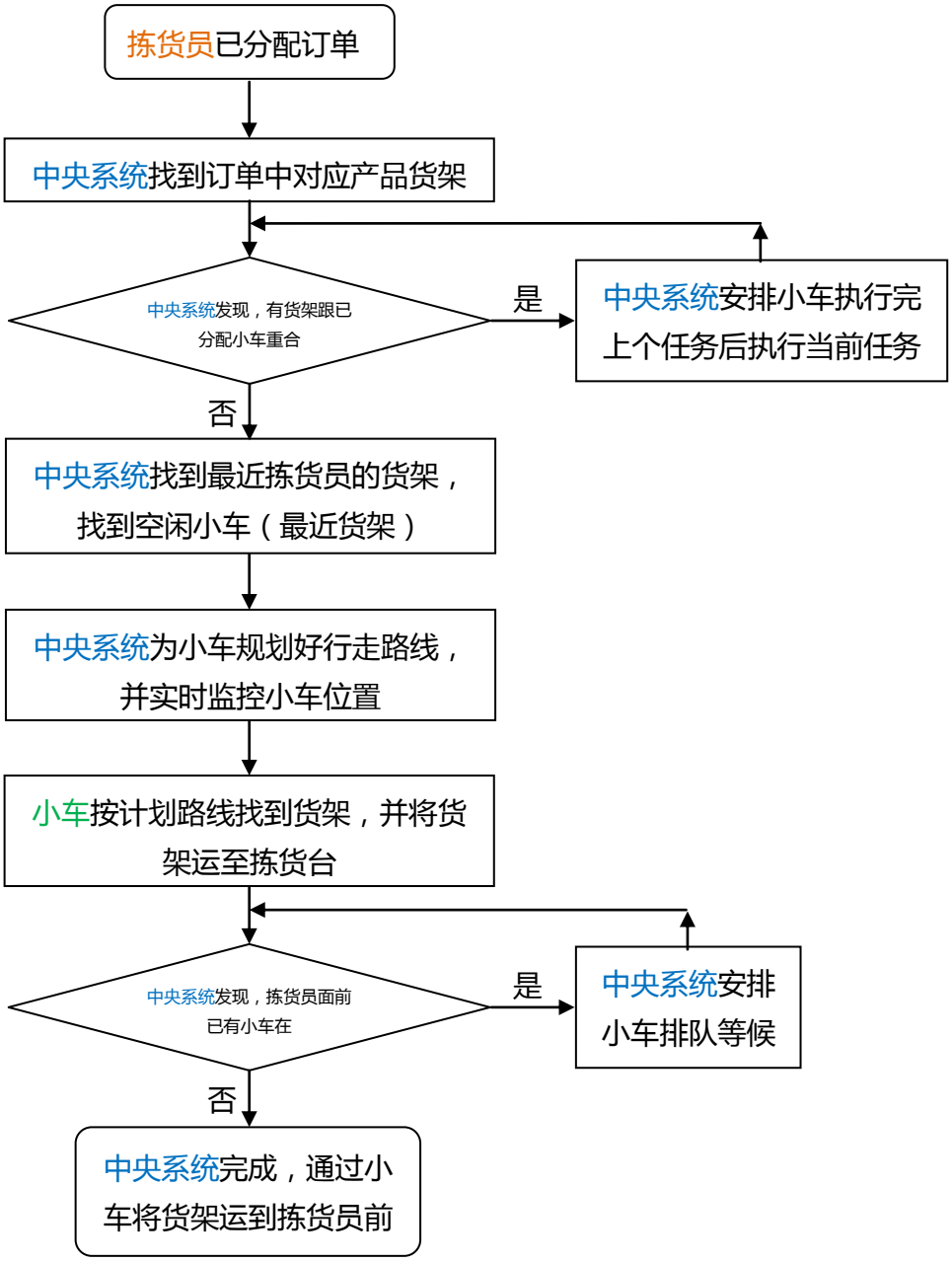
# 系统工作流程图

**拣货**（暂不考虑通过设备运送订单货架，仅通过人工选择订单容器）

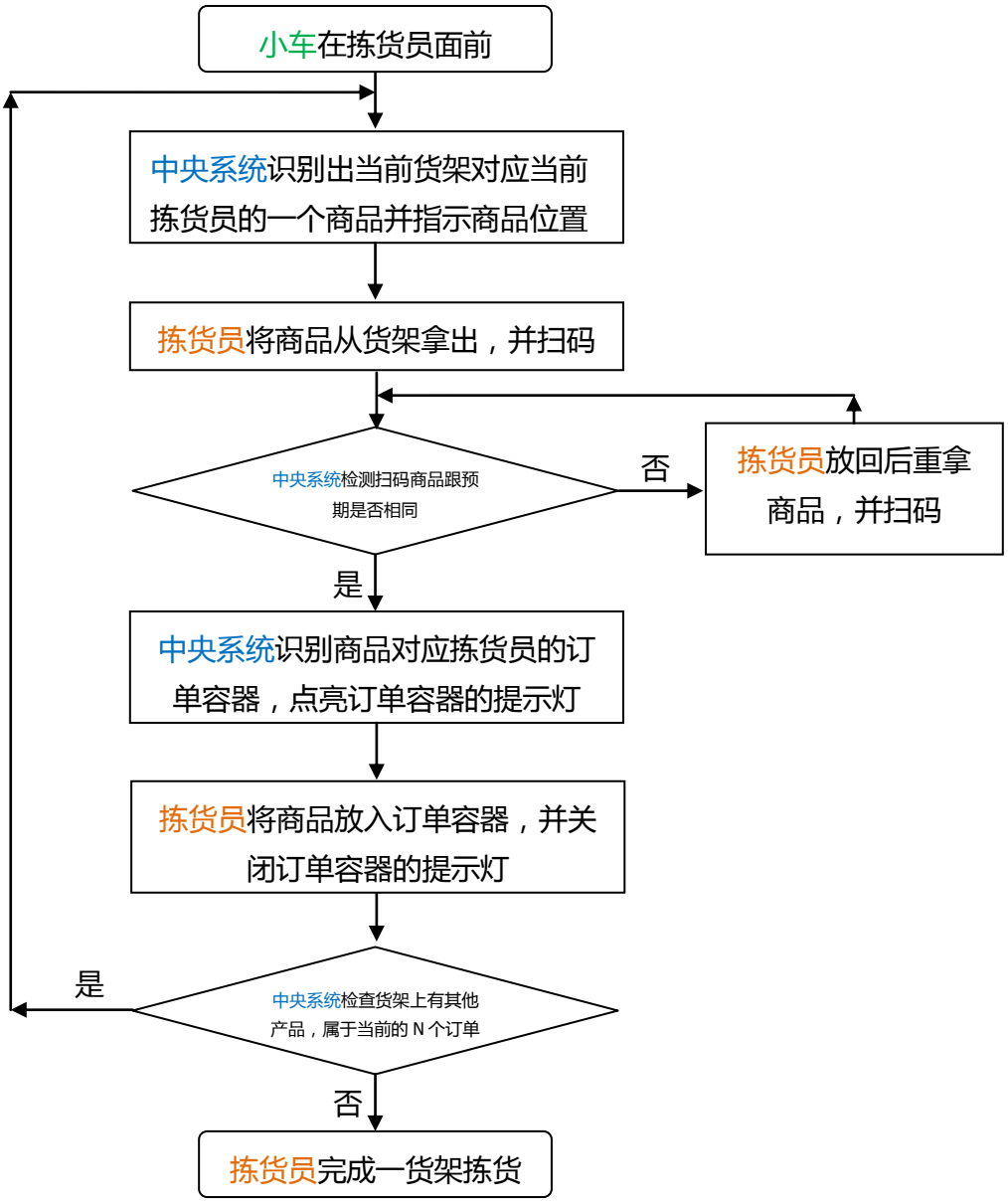
**订单分配给拣货员**



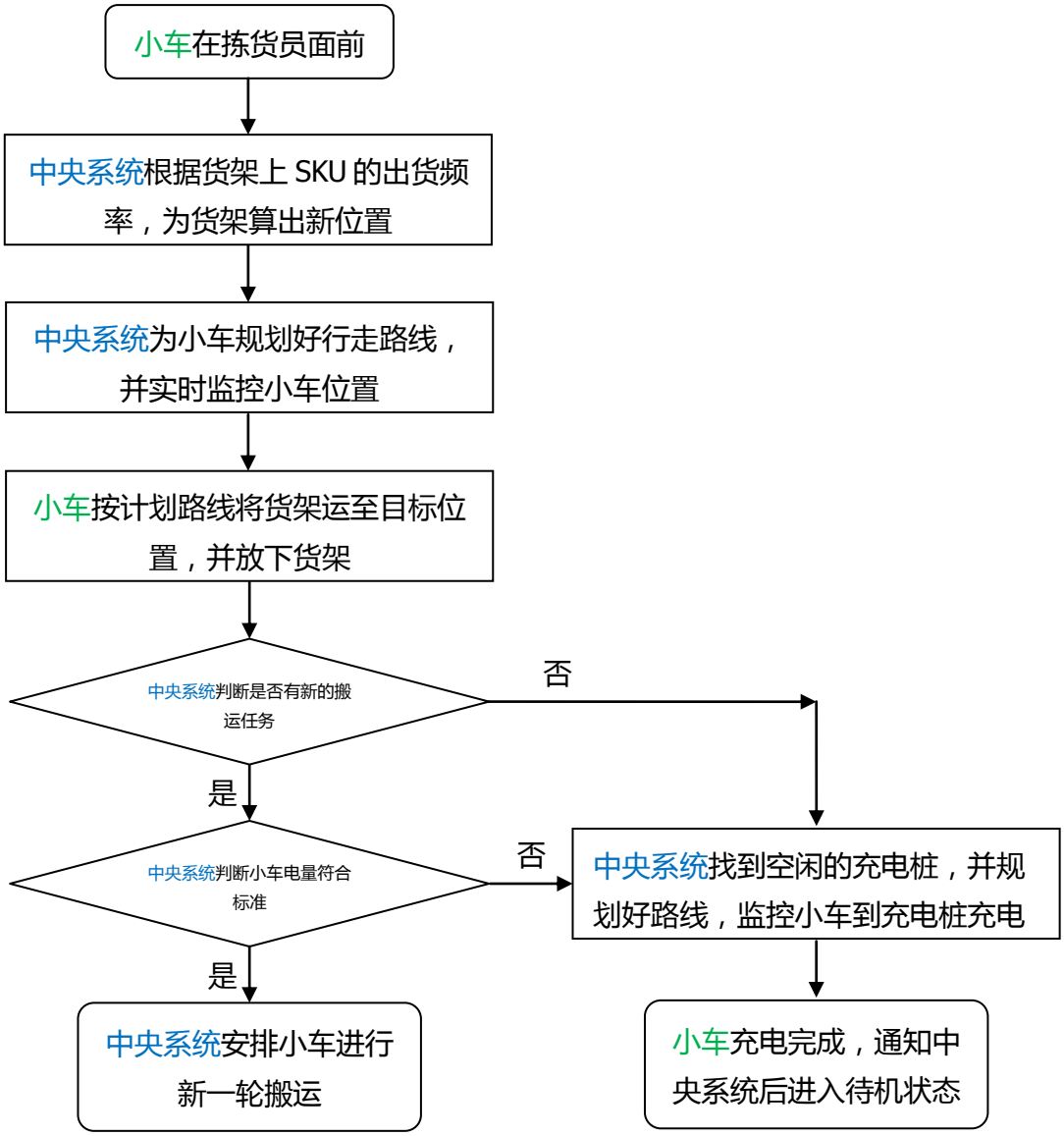

**小车运货架到拣货员面前**



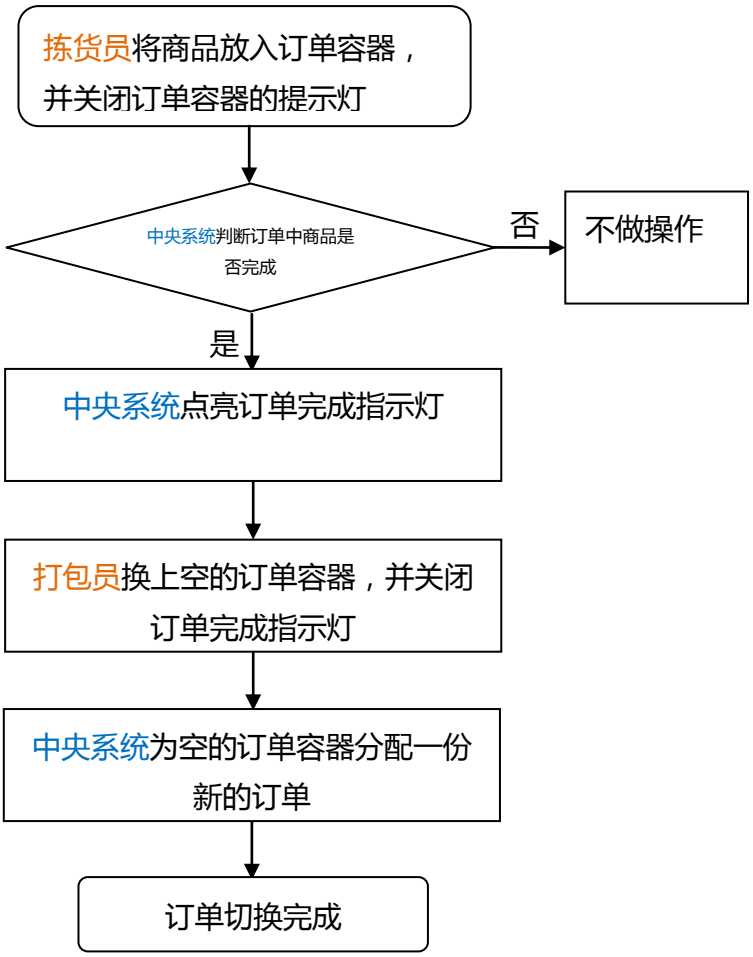
**拣货员拣一个货架上的货**



**小车将货架运回仓储区**


小车在拣货员面前
中央系统根据货架上 SKU 的出货频率，为货架算出新位置
中央系统为小车规划好行走路线，并实时监控小车位置
小车按计划路线将货架运至目标位置，并放下货架
中央系统判断是否有新的搬运任务
否
是
中央系统判断小车电量符合标准
否
中央系统找到空闲的充电桩，并规划好路线，监控小车到充电桩充电
是
中央系统安排小车进行新一轮搬运
小车充电完成，通知中央系统后进入待机状态

**订单切换**

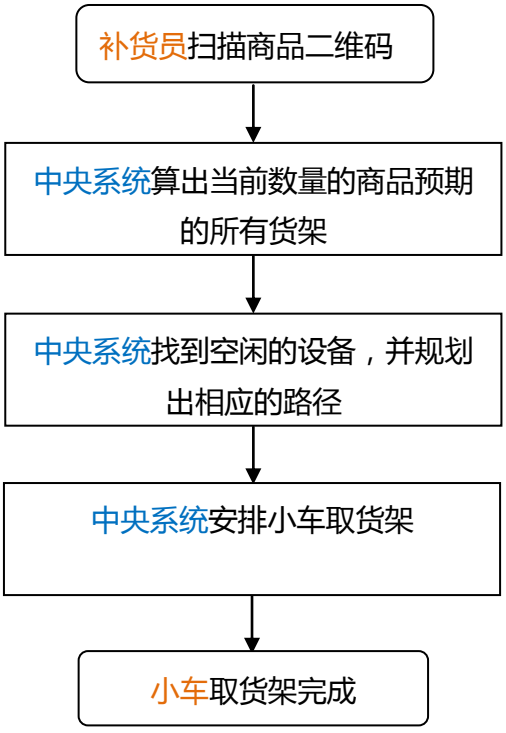

拣货员将商品放入订单容器，并关闭订单容器的提示灯
中央系统判断订单中商品是否完成
否
不做操作
是
中央系统点亮订单完成指示灯
打包员换上空的订单容器，并关闭订单完成指示灯
中央系统为空的订单容器分配一份新的订单
订单切换完成

## 补货

补货应该是先给出补货计划，补货计划是对于订单的预期以及当前库存量得到的，所以，补货商品是在库存中有预期的对应库位的。

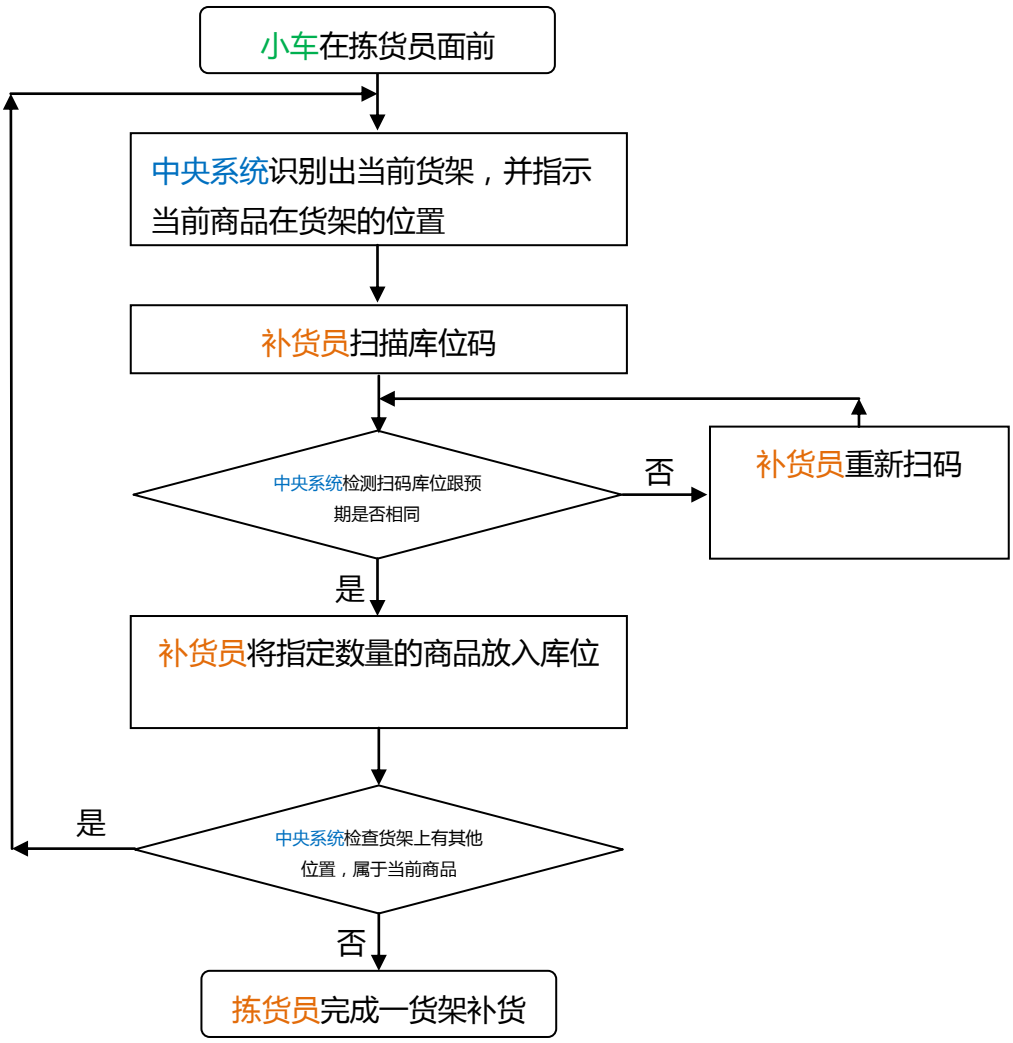
操作是拣货的逆向过程，区别有两点：1，定位货架的查询条件不同：当补货员扫描商品后，根据要补货的数量和货架自身空位量，以及最近一周跟补货商品的销售相似度，综合考虑来决定为哪个货架进行补货，从而选择哪个货架；而拣货需要是订单中的产品，所以只需要考虑，订单需要移动的货架数量、商品先进先出和距离拣货员的距离作为考虑因素。2，货架往返的执行结果不同：对于要补货的货架是上架商品，拣货是相反的下架商品。

**补货员对商品扫码，小车取货架**




**小车运货架到补货员面前（同拣货）**

**补货员将商品放到货架**




**小车将货架运回仓储区（同拣货）**

**商品切换**

补货员换个商品进行扫码

**盘点**

盘点操作跟拣货类似，两个区别分别是：1，对于货架的选择条件，此处是随机一个货架；2，没有对货物的上下架，仅检查核实，操作之后没有商品的增减。

**按货架盘点**

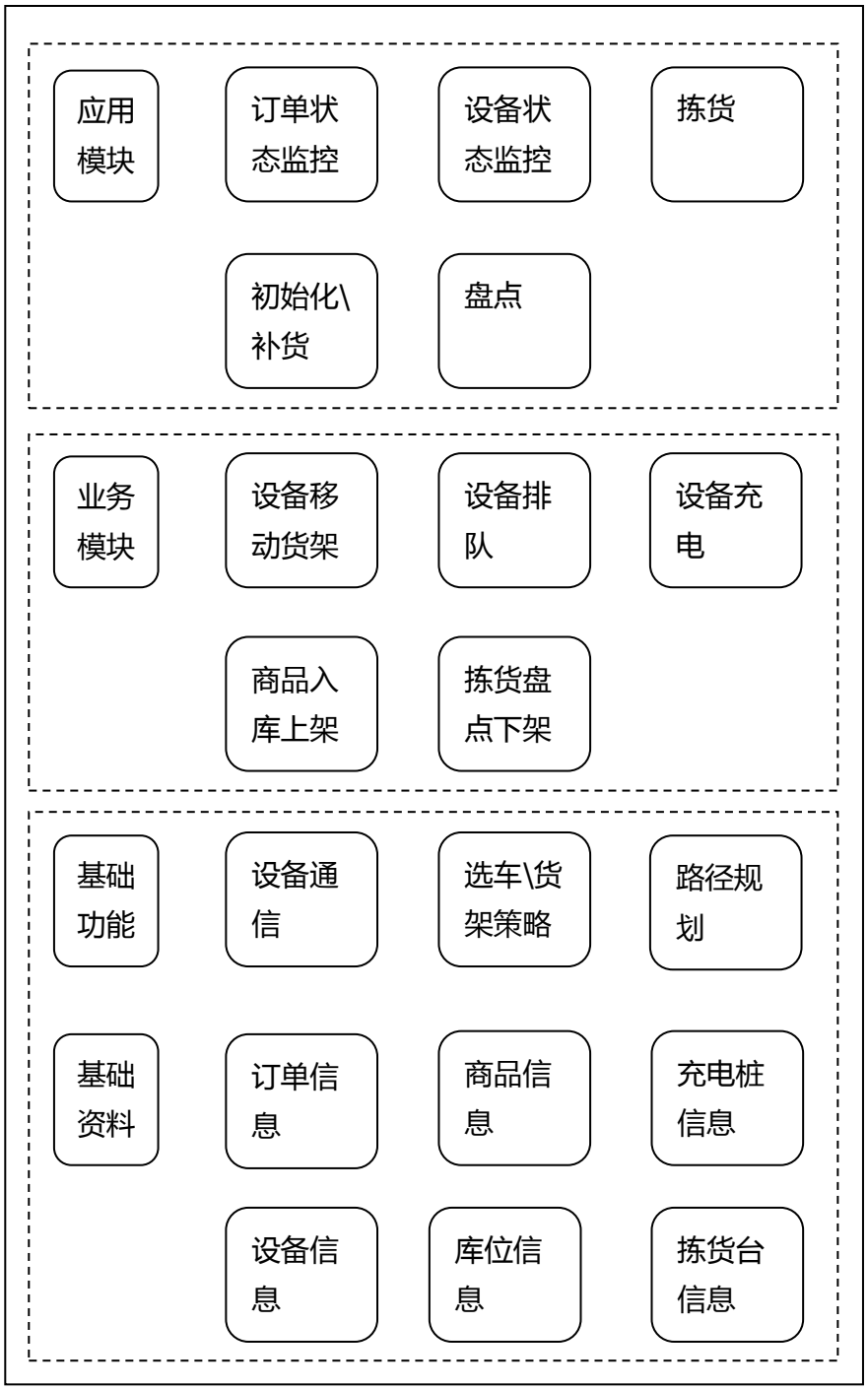
随机选中一台货架，给出系统中货架上的产品及数量，由盘点员确认对应的实际结果。

**按商品盘点**

随机选中一款商品，给出系统中的货架上的产品及数量，有盘点员确认对应的实际结果。

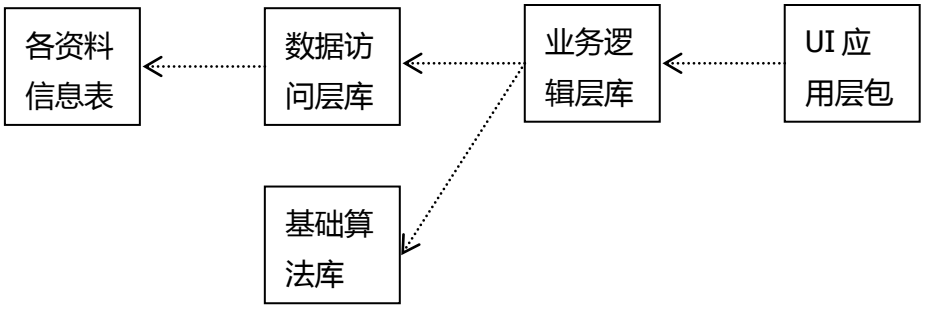
**按拣货员/订单盘点**

## 软件模块图

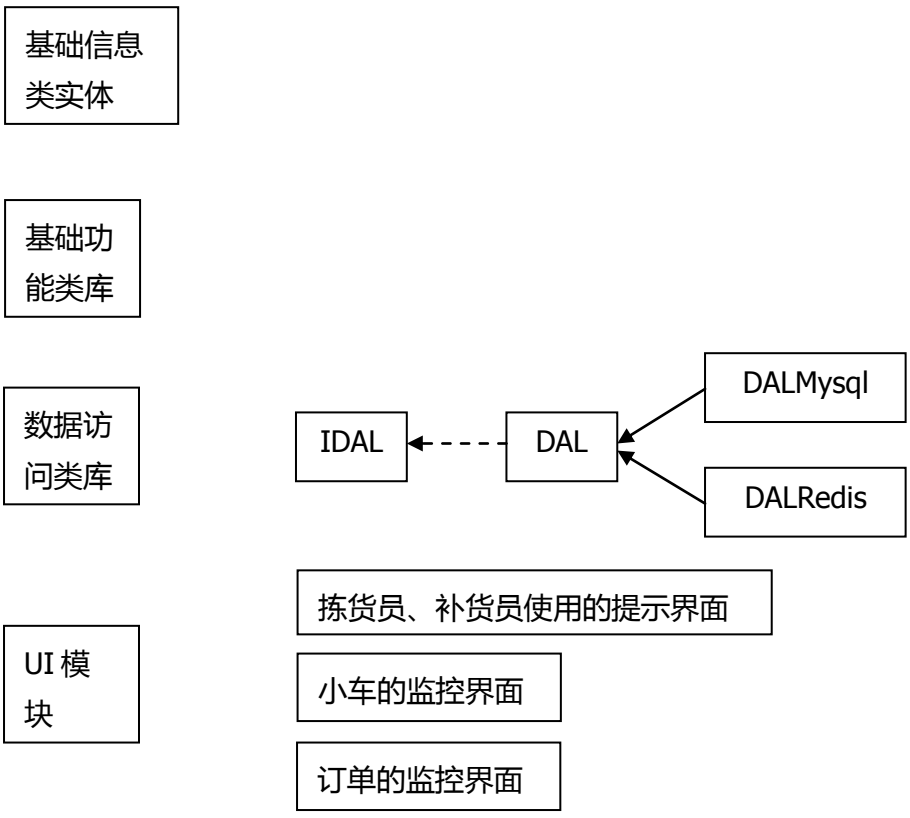

应用模块
订单状态监控
设备状态监控
拣货
初始化\补货
盘点
业务模块
设备移动货架
设备排队
设备充电
商品入库上架
拣货盘点下架
基础功能
设备通信
选车\货架策略
路径规划
基础资料
订单信息
商品信息
充电桩信息
设备信息
库位信息
拣货台信息

## 软件 UML 图

### 包图


各资料信息表
数据访问层库
业务逻辑层库
UI 应用层包
基础算法库

### 类图


基础信息类实体
基础功能类库
数据访问类库
IDAL
DAL
DALMysql
DALRedis
UI 模块
拣货员、补货员使用的提示界面
小车的监控界面
订单的监控界面

## 数据表结构

**持久记录**（用于初始化）

| 拣货员表 |
| --- |
| 人员 ID：1 |
| 姓名：张三 |
| 性别：男 |
| 权限：11 |
| 职位：管理员 |
| 年龄：20 |
| 手机：150XXX |
| 联系地址：南山蛇口 |

| 货架信息表 |
| --- |
| 货架 ID：1 |
| 货架编码(6 位：仓库 2+行数 1+列数 2+层数 1)：02A271 |
| 坐标索引：2 |
| 位置历史：3，2 |
| 货架层数：4 |
| 货架面数：2 |
| 各面（分号分隔）每层格数：01020201; 01020301 |
| 类型（冗余字段：1 小 2 中 3 大 11 冷）：1 |

| SKU 信息表 |
| --- |
| SKU ID：1 |
| 商品名称：水杯 |
| 库存数量：3 |
| 型号：300ml |
| 颜色：红色 |
| 尺寸 (mm)：20*200*2000 |
| 重量(g)：200 |
| 备注信息：易碎 |

| 商品信息表（bak） |
| --- |

| |
|---|
| 商品 ID(通过条码/ID 唯一定位)：1 |
| 条码编号：DE34553233 |
| SKU ID：1 |
| 货架 ID：1 |
| 货架面号（最多 4 个面）：1 |
| 库位号（当前货架所有库位自下到上，自左到右编号）：2 {此库位仅用于显示给拣货员：若将库位，层号信息分别用数据记录表示，数据冗余不增加查询和计算效率} |
| 商品名称及规格（用于显示）：水杯；红色 300ml |
| 生产日期：2015-07-01 |
| 过期日期：2016-12-31 |
| 尺寸规格(mm)：20*200*2000 |
| 重量(g)：200 |
| 数量：2 |
| 上架时间（扫码/指派）：2016-07-01 10:51:50 |
| 出库时间（扫码/指派）：2016-07-10 10:51:50 |
| 商品状态（0 正常 3 部分出库 9 出库）：0 |

| **设备信息表** |
|---|
| 设备 ID：1 |
| 设备序号（可读编号）：S0012 |
| 状态（0 待命 3 工作中 8 充电中 9 故障）：1 |
| 绝对坐标 X，Y，Z：（2，3，1） |
| IP 地址：192.168.1.111 |
| 厂家：XX 有限公司 |
| 生产日期：2016-07-17 |
| 上线日期：2016-07-19 |
| 备注：哈哈哈 |

| **订单信息表** |
|---|
| 自增 ID：1 |
| 订单编号：SD0012 |
| SKU 信息：1,2;4,1 |
| 商品总数：3 |
| 优先级：1 |

| |
|---|
| 状态（0 未处理， 5 已处理，9 异常）：0 |
| 拣货台 ID：1 |
| 拣货台 ID：1 |
| 导入订单/下单时间：2016-07-15 10:10:10 |
| 开始拣货时间：2016-07-15 11:10:10 |
| 备注：哈哈哈 |

| **仓库内站点（补货/拣货台/充电桩）** |
|---|
| 序号 ID：1 |
| 可读序号：HT002 |
| 仓库 ID：1 |
| IP 地址：192.168.1.10 |
| 位置索引：1 |
| 绝对坐标 X，Y，Z：（2，3，1） |
| 类型（5 补货 3 拣货 2 充电桩）：2 |
| 状态（0 空闲 3 工作中 9 故障）：3 |
| 备注： |

| **仓库内位置坐标表** |
|---|
| 位置 ID：1 |
| 仓库序号：2 |
| 绝对坐标 X，Y，Z：（2，3，1） |
| 状态（0 正常 3 工作中 8 阻塞 9 故障）：0 |
| 类型（1 货架点 2 拣货台 3 补货台 4 充电桩 5 路交叉点）：1 |

| **仓库内路线表** |
|---|
| 序号 ID：1 |
| 仓库序号：2 |
| 位置 1 ID：1 |
| 位置 2 ID：2 |
| 类型（1 正向 2 反响 3 双向）：1 |
| 状态（0 正常 8 阻塞 9 故障）：0 |

**实时状态记录**（用于跟踪调试 - Mysql）

| 实时订单表<br>订单开始拣货时增加记录，每次录入商品更新记录 |
| --- |
| 自增序号：1 |
| 订单 ID：5542144 |
| 商品总数：5 |
| 商品信息（SKU ID，数量）：1,2;3,1;4,1;6,1 |
| 拣货员 ID：3 |
| 拣货台 ID ：1 |
| 取货设备（冗余字段：设备编号）：1,2 |
| 取货商品（冗余字段：SKU ID，条码 ID，数量;）：1,1,1;1,2,1;3,3,1 |
| 已取数量：3 |
| 状态（0 未处理，3 拣货中，5 完成拣货，9 异常）：3 |
| 拣货备注：2016-07-20 10:10:10（2），2016-07-20 10:13:10（1） |

| 实时拣货商品表<br>员工每次开始为一个订单拣货，增加一个订单的所有商品记录 |
| --- |
| 自增序号：1 |
| 站台 ID:1 |
| 订单 ID：1 |
| SKU ID：11 |
| 总数量：2 |
| 已完成数量：1 |
| 状态（0 未处理，3 处理中，5 已完成，9 异常）：3 |
| 最后更新时间：2016-07-01 13:50:10 |

| 实时移动货架表<br>给设备发布取货任务时增加记录，小车业务状态改变后更新状态 |
| --- |
| 自增序号：1 |
| 货架 ID：1 |
| 设备 ID：1 |
| 商品数量：1 |
| 商品 ID（通过条码/ID 唯一定位）：1,2,3 |
| SKU ID（同个 SKU 包含多个商品）：1,2,3 |
| 订单 ID：1,2,3 |
| 拣货台 ID：1,2,3 |
| 状态（3 取货中，5 已完成，9 异常）：3 |

| |
|---|
| 开始取货时间：2016-07-01 13:50:10 |
| 到达货架时间：2016-07-01 13:59:10 |
| 搬起货架时间：2016-07-01 13:59:10 |
| 送达货架时间：2016-07-01 13:59:10 |
| 完成拣货时间：2016-07-01 13:59:10 |
| 送回货架时间：2016-07-01 13:59:10 |

| **实时设备位置表**<br>设备连接后，每秒增加 5 条记录（前期先用 Mysql，吞吐支持不到时再独立） |
|---|
| 自增序号：1 |
| 设备 ID：1 |
| 状态（0 候命中 1 取货中 2 运货中 3 电量低 4 充电中 11 故障 12 失联）：1 |
| 当前功能：1 |
| 起点位置索引：1 |
| 终点位置索引：2 |
| 当前位置索引：3 |
| 当前绝对位置 X，Y，Z： 1，2，3 |
| IP 地址：192.168.1.111 |
| 当前时间：2016-07-01 13:50:10 |
| 备注：电量低时记录电量，取货中时记录商品名称，运货中时记录货架编号 |

**日志、异常记录**每步操作（用于历史备案 - 文档/Nosql）

| **通讯记录表** |
|---|
| 记录时间：2016-07-01 11:20:50 |
| 对象类型（1 设备 2 拣货员）：1 |
| 行为主体 ID：1 |
| 操作步骤（功能名称）：设备取货 |
| 日志内容（操作记录）：设备接收取货指令（XXX） |

| **员工行为记录（拣货/补货/登入/登出）表**<br>员工每次扫码拣货、补货后增加一条记录 |
|---|

| |
|---|
| 自增序号：1 |
| 员工 ID：1 |
| 行为类型（1 拣货 2 补货 3 上班 4 下班）：1 |
| 商品 ID：11 |
| SKU ID：11 |
| 所在站台 ID：22 |
| 操作时间：2016-07-01 13:50:10 |

| **异常记录表** |
|---|
| 记录时间：2016-07-01 11:20:50 |
| 对象类型（1 设备 2 拣货员）：1 |
| 行为主体 ID：1 |
| 操作步骤：设备取货 |
| 异常信息：发信息（XXX）获取设备（XXX）状态，响应超时 |

**状态信息解释**

**动态状态** -- 行为结果（0 未处理，3 处理中，5 已完成，9 异常）

实时订单状态（0 未处理，3 拣货中，5 完成拣货，9 异常）

订单状态（0 未处理， 5 已处理，9 异常）

实时拣货商品状态（0 未处理，3 处理中，5 已完成，9 异常）

实时移动货架状态（3 取货中，5 已完成，9 异常）

**静态状态** -- 自身状态（0 正常 3 工作中 5 已完成 8 阻塞 9 故障）

仓库内位置坐标表状态（0 正常 3 工作中 8 未工作 9 故障）

仓库内路线状态（0 正常 8 阻塞 9 故障）

仓库内站点状态（0 空闲 3 工作中 9 故障）

设备信息状态（0 待命 3 工作中 8 充电中 9 故障）

商品信息状态（0 正常 3 部分出库 5 出库)

**其他状态**

**故障状态**

# 通讯协议

## 通信协议格式

| | |
|---|---|
| 开始位 | < |
| 包数据高字节 | 0 |
| 包数据低字节 | N1 |
| 保留位高字节 | 头文件属性 |
| 保留位低字节 | 头文件属性 |
| 功能码 1 | 0x01 |
| 功能 1 数据高字节 | 0 |
| 功能 1 数据低字节 | N1 |
| 功能 2、3 | 功能保留（6 位） |
| 功能码 4 | 0x04 |
| 功能 4 数据高字节 | 0 |
| 功能 4 数据低字节 | N4 |
| 数据位 1 | 0xAF |
| 数据位 2 | 0xAF |
| …… | …… |
| 数据位 N1 | 0xAF |
| 若有其他功能的数据则依次顺序排序，若没有则不保留位置（包括功能 1） | |
| 校验码 | 0xAF |
| 校验码 | 0xAF |

## 头文件属性

| 属性名称 | 所在位置 | 参数说明 |
|---|---|---|
| 是否需要回复 | 低字节低位第 2 位<br>0000 0000 0000 0010 | 0：不需要回执<br>1：需要 |
| | | |
| | | |

## 功能码

| 功能名称 | 功能码 | 数据位数据格式 |
|---|---|---|
| 上位机：查询状态 | 0x10 | |
| 上位机： | 0x11 | |

| | | |
|---|---|---|
| 回执 | | |
| 上位机：停止移动 | 0x12 | |
| 上位机：转向 | 0x13 | 顺 0/逆 1 旋转（1Byte），转动次数（1Byte） |
| 上位机：安排充电 | 0x20 | 充电桩 ID（2 Byte），第一步 X（2 Byte），第一步 Y（2 Byte），第一步 Z（1 Byte），……,第 i 步 X（2 Byte），第 i 步 Y（2 Byte），第 i 步 Z（1 Byte）……，最后一步 X（2 Byte），最后一步 Y（2 Byte），最后一步 Z（1 Byte） |
| 上位机：移动到位置等待 | 0x21 | 保留位（2Byte）,第一步 X（2 Byte），第一步 Y（2 Byte），第一步 Z（1 Byte），……,第 i 步 X（2 Byte），第 i 步 Y（2 Byte），第 i 步 Z（1 Byte）……，最后一步 X（2 Byte），最后一步 Y（2 Byte），最后一步 Z（1 Byte） |
| 上位机：去找货架 | 0x22 | 货架 ID（2 位），第一步 X（2 Byte），第一步 Y（2 Byte），第一步 Z（1 Byte），……,第 i 步 X（2 Byte），第 i 步 Y（2 Byte），第 i 步 Z（1 Byte）……，最后一步 X（2 Byte），最后一步 Y（2 Byte），最后一步 Z（1 Byte） |
| 上位机：运货架到拣货台 | 0x23 | 拣货台 ID（2 位），第一步 X（2 Byte），第一步 Y（2 Byte），第一步 Z（1 Byte），……,第 i 步 X（2 Byte），第 i 步 Y（2 Byte），第 i 步 Z（1 Byte）……，最后一步 X（2 Byte），最后一步 Y（2 Byte），最后一步 Z（1 Byte） |
| 上位机：送回货架到仓储区 | 0x24 | 货架 ID（2 Byte），第一步 X（2 Byte），第一步 Y（2 Byte），第一步 Z（1 Byte），……,第 i 步 X（2 Byte），第 i 步 Y（2 Byte），第 i 步 Z（1 Byte）……，最后一步 X（2 Byte），最后一步 Y（2 Byte），最后一步 Z（1 Byte） |
| 小车：当前状态/心跳 | 0x30 | ~~工作状态（1 Byte）{0 闲置，2 充电，3 取货架，4 送拣货台，5 排队，6 归位货架，1 故障}，电量（1 Byte）{百分比值}~~，小车 ID（2 Byte），X 坐标（2 Byte），Y 坐标（2 Byte），Z 坐标（1 Byte） |
| 小车：电量低 | 0x31 | 电量（2 Byte）{百分比值），X 坐标（2 Byte），Y 坐标（2 Byte），Z 坐标（1 Byte） |
| 小车：遇到障碍 | 0x32 | 障碍距离（2 Byte），X 坐标（2 Byte），Y 坐标（2 Byte），Z 坐标（1 Byte） |
| 小车：超载 | 0x33 | 货物重量（2 Byte），X 坐标（2 Byte），Y 坐标（2 Byte），Z 坐标（1 Byte） |
| 小车： | 0x34 | 货物重量（2 Byte），X 坐标（2 Byte），Y 坐标（2 Byte），Z |

| | | |
|---|---|---|
| 货物不稳 | | 坐标（1 Byte） |
| 小车：未知异常 | 0x39 | 保留位（2Byte）,X 坐标（2 Byte），Y 坐标（2 Byte），Z 坐标（1 Byte） |
| 小车：找到货架并抬起 | 0x41 | 小车 ID（2Byte）,X 坐标（2 Byte），Y 坐标（2 Byte），Z 坐标（1 Byte） |
| 小车：到拣货台 | 0x42 | 小车 ID（2Byte）,X 坐标（2 Byte），Y 坐标（2 Byte），Z 坐标（1 Byte） |
| 小车：送回货架并放下 | 0x43 | 小车 ID（2Byte）,X 坐标（2 Byte），Y 坐标（2 Byte），Z 坐标（1 Byte） |
| 小车：收到取货架命令 | 0x44 | 小车 ID（2Byte） |
| 拣货员：请求订单 | 0x50 | 拣货员 ID（2 Byte），拣货台 ID（2Byte），订单数量（2Byte），保留位（1Byte） |
| 拣货员：找到商品 | 0x51 | 拣货台 ID（2Byte）, 商品条码(10 Byte)， 商品数量（2Byte），保留位（3Byte） |
| 拣货员：商品放入订单分拣 | 0x52 | 货架 ID（2Byte），订单 ID（2Byte），商品 ID（2Byte）, 保留位（1Byte） |
| 上位机：分配订单 | 0x61 | 1 是 2 否初次分配（2Byte），第一张订单 ID（2Byte）,第一订单商品总数（2Byte）,保留位（1Byte），……，第 i 张订单 ID（2Byte）,第 i 订单商品总数（2Byte）,保留位（1Byte），……, 最后一张订单 ID（2Byte）, 最后一订单商品总数（2Byte）,保留位（1Byte） |
| 上位机：当前商品对应订单 | 0x62 | 订单 ID（2Byte），商品 ID（2Byte），Sku ID（2Byte），保留位（1Byte） |
| 上位机：货架待拣商品对应信息 | 0x63 | 货架 ID（2Byte），商品库位（2Byte），商品 ID（2Byte）,保留位（1Byte），货架库位类型（20Byte），商品名称（30Byte）， |
| 上位机：拣货结果 | 0x64 | 0 成 1 败（2Byte）,失败原因（30Byte） |

## 路径规划

直接按坐标接近来决定行走方向，转弯越少越好

**选择策略**（移动总曼哈顿距离最短）

选订单：

初始订单时，按时间前 N 个(选择有更多共同商品，或者在相同货架的订单)；

换新订单时，最早订单（选择跟当前拣货员待拣商品更多重合的订单）

选货架：

拣货时，选择拥有更多待检订单商品、靠近拣货台的货架

补货时，选择跟对应 SKU 出货频率接近的货架，同时满足更接近（大于）当前商品尺寸空库位的货架

选小车：

最靠近货架的空闲小车

## 算法实现逻辑

绿色背景为当前方案

### 计算待选货架

Si：第 i 个货架，Gi：第 i 个商品 SKU，Gic：第 i 个商品有 c 个

1，当前拣货员 =》 所有订单

=》所有商品 SKU 及 数量

=》 所有商品货架 及 对应货架上商品数量

=》按商品 SKU 分类取得所有货架组合

=》计算每个商品满足数量的货架组合

=》货架原子集（用小单元组合）

=》从每个商品的组合中挑选一个重新组合，并去重

=》货架原子集（用小单元组合）

G1：S1+S2, S2+S3, S1+S3

G2：S1, S3

G3：S2, S3

G4：S2, S4

2，当前拣货员 =》 所有订单

=》所有商品 SKU 及 数量

=》 所有商品货架 及 对应货架上商品数量

=》所有货架组合

=》计算所有商品满足数量的货架组合

=》货架原子集（用小单元组合）

S1：G1c, G2c

S2：G1c, G3c, G4c

S3：G1c, G2c, G3c

S4：G4c

### 确定货架

获得货架集合

=》集合所有货架到拣货台的总距离

=》最小距离，相同距离用最少货架，货架数相同随机取

**路径搜索**

数据结构的复杂度

V：节点数；E：路径数；Degree(V)：节点维度（V 节点路径数）

| 结构名称 | 空间复杂度 | 时间复杂度 | | |
|---|---|---|---|---|
| | | 增加一条路径 | 相邻节点的路径 | 遍历一个节点的路径 |
| 路径 | E | 1 | E | E |
| 邻接矩阵 | $V^2$ | 1 | 1 | V |
| 邻接链表 | E+V | 1 | Degree(V) | Degree(V) |

由于仓库的结构中每个节点仅跟附近节点有路径，大部分节点之间没有连接，使用邻接矩阵将太多空间闲置，路径的查询效率相对有点低，暂时考虑使用邻接链表实现。

| 下标 | 数据 | 节点位置 | 名称 | 边指针列表 |
|---|---|---|---|---|
| 1 | 22 | 22，33，11 | AA | 索引 权重 距离：2 1 1；索引 权重 距离：3 1 1；索引 权重 距离：4 1 1 |
| 2 | 33 | 22，43，11 | AB | 索引 权重 距离：1 1 1；索引 权重 距离：3 1 1 |
| 3 | 44 | 22，53，11 | AC | 索引 权重 距离：1 1 1；索引 权重 距离：2 1 1 |
| 4 | 55 | 32，53，11 | CA | 索引 权重 距离：1 1 1 |
| 5 | 45 | 33，53，11 | CB | |
| 6 | | | | |
| | | | | |

搜索算法

1. 深度优先搜索：沿一条路纵向搜索
2. 广度优先搜索：按层次横向搜索

最短路径算法

1. Dijkstar：从未访问节点中选择距离出发点最短路径的节点，直到所有节点被访，数据结构用邻接链表

=》结果是两点间的最短路径，若需要计算所有节点间距离也可以，则需要为每个节点都执行当前算法，整体时间复杂度跟 floyd 算法一致，但 floyd 算法的计算量小。

**在未访问节点中寻找最短路径节点的算法逻辑：**

**初始代码逻辑**：外层循环遍历未访问节点列表，内层循环遍历已访问节点列表，依次判断两个循环中节点间的最短距离，两个列表节点数和固定，时间复杂度为 O（$n^2$）=n*(n-m)，m,n 分别为未访问和全部节点数

```
int minIdx = 0, minDistance = Int32.MaxValue, tmpLen;
foreach (int item in UnkownList)
{
    foreach (KeyValuePair<int, int> visitor in VisitedList)
    {
        tmpLen = graph.CheckEdgeDistance(item, visitor.Key);
        if (tmpLen > 0 && tmpLen + visitor.Value < minDistance)
        {
            minIdx = item;
            minDistance = visitor.Value + tmpLen;
        }
    }
}
pathList.Add(graph.GetHeadNodeByID(minIdx));
VisitedList.Add(minIdx, minDistance);
UnkownList.Remove(minIdx);
```

**优化后的逻辑**：先在未访问节点列表中循环，找到跟起点建立联系节点的最短距离，再循环更新未访问节点中通过所选节点被缩短的距离，时间复杂度为 O（n）=2*m， m 为未访问节点数（未访问节点会越来越少）

```
int minIdx = 0, minDistance = Int32.MaxValue, tmpLen;
//先找到当前最小的距离值
foreach (KeyValuePair<int, int> item in UnkownList)
{
    if (item.Value > 0 && item.Value < minDistance)
    {
        minIdx = item.Key;
        minDistance = item.Value;
    }
}
pathList.Add(graph.GetHeadNodeByID(minIdx));
VisitedList.Add(minIdx, minDistance);
//再更新新节点减少的距离
foreach (KeyValuePair<int, int> item in UnkownList)
{
    tmpLen = graph.CheckEdgeDistance(item.Value, minIdx);
    if (tmpLen > 0 && (item.Value < 0 || tmpLen + minDistance < item.Value))
    {
        UnkownList[item.Key] = tmpLen + minDistance;
    }
}
```

2. Floyd-Warshall：计算每个节点（A）对任意两点间（X、Y）距离的贡献，若 XY 两点间距离因节点 A 而缩短，则用 A 作为 XY 间路线的一站，数据结构用邻接矩阵

   =》结果是任意两点间的最短距离，优点是一次计算一直使用，缺点是计算的时间复杂度较高（$O(V^3)$）

3. 最终算法（两种都实现，分别用于不同场景）

   a) 默认使用第二种算法，调度时只需知道起止点则不需要计算路径，可以直接使用初始计算结果

   b) 当设备遇到故障或者遇到堵塞时，需要重新规划路线时，将某个位置设为不可行，通过第一种方案重新计算路径。

## 待确定问题：

1. 对接原 OMS/WMS 数据（订单、产品）的方案

   关键点：

   - 订单信息：仅需要可以定位到商品即可（商品 ID），需要 OMS 提供接口获取订单对应的产品（通常商家可能不愿意提供订单的客户信息）
   - 产品所在库位信息：跟 WMS 系统绑定，或者用 WMS 中产品信息进行初始化
   - 库位、拣货台、充电桩及设备信息：独立导入/录入

2. 数据存储方案

   关键点：

   - 基础信息存储在持久数据库 Mysql 中（数据格式相对固定不考虑 Nosql，收费不考虑 Sqlserver，选择 Mysql）
   - 多个小车位置状态改变频繁，每秒更新 5 次小车位置，Mysql 可以胜任 100 台左右小车，若更多小车，则需要引入实时数据库 ExtremeDB（会有高并发不考虑内存数据库 Sqlite）

3. 对设备的路径规划及运动控制是设备独立实现，不采用中心调度

4. 入库/补货阶段如何通过商品二维码跟商品 SKU 做绑定