

中国高校计算机大赛 网络技术挑战赛

基于 AIoT 和区块链的智慧快递运输 管理系统

作品设计文档

所在赛道与赛项：A 赛道

2023 年 8 月

目录

第 1 章 目标问题与意义价值	1
1.1 项目背景	1
1.2 存在问题	2
1.3 系统功能	3
1.4 意义价值	3
第 2 章 设计思路与方案	5
2.1 系统总体设计	5
2.2 感知层/网关层设计	6
2.3 云层设计	8
2.3.1 数据库设计	8
2.3.2 路径规划算法设计	11
2.3.3 联盟链设计	15
2.4 应用层设计	18
2.4.1 网页端	18
2.4.2 小程序端	19
第 3 章 方案实现	21
3.1 开发环境要求	21
3.2 感知层/网关层实现	21
3.2.1 数据采集	21
3.2.2 多传感器数据融合	23
3.2.3 数据处理	25
3.2.4 网关数据上传	25
3.3 云层实现	26
3.3.1 疲劳驾驶检测	26
3.3.2 改进遗传-蚁群融合算法实现多点路径规划	27
3.3.3 信息可追溯	31
3.4 应用层实现	34
3.4.1 网页 Web 实现	34
3.4.2 小程序实现	34
第 4 章 应用效果	35
4.1 数据采集上传实现效果	35
4.2 疲劳驾驶识别实现效果	37
4.3 多目的地路径规划实现效果	38
4.4 快递信息可追溯实现效果	39

4.5 前端 Web 实现效果	41
4.6 小程序实现效果	47
第 5 章 创新与特色	50
5.1 改进的遗传-蚁群融合算法	50
5.2 基于层次身份的加密和环签名	50
5.3 快递运输过程实时监控	51
第 6 章 未来展望	52
6.1 作品成果	52
6.2 应用前景	52

第 1 章 目标问题与意义价值

1.1 项目背景

随着我国经济与电子商务的不断发展，网络购物逐步成为了居民的一种流行的购物方式。伴随着电商和网购的发展，快递物流需求也增长十分迅速。由图 1-1 近 10 年的中国快递业务量统计数据显示，快递业务量规模在短短几年的时间已经翻了十余倍。快递物流需求持续不断地飞速增长，我国的快递业务量已连续 8 年位居世界第一。

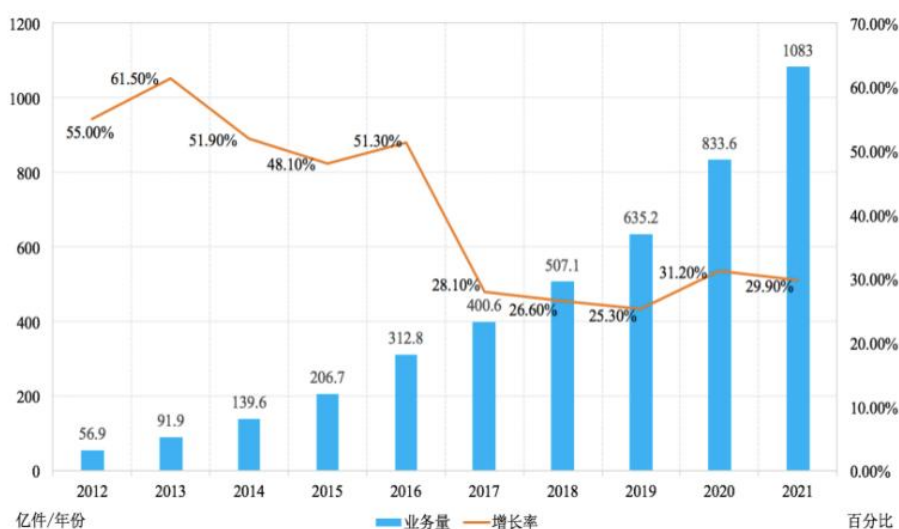


图 1-1 中国快递业务量增长趋势

然而，在快递业务快速发展的同时，也产生了许多新挑战。**运输安全性差、配送效率低、用户隐私泄露、过程不透明**等问题制约了行业的发展，同时也损害了消费者的权益。较低的快递物流服务水平会降低消费者服务满意度，影响其购物体验，长期以往将给快递物流企业带来巨大的损失。如何进一步改善快递物流服务水平，提高快递物流服务质量对于快递业的发展十分重要。

随着互联网技术、物联网技术和人工智能技术的不断发展，物流行业也在不断变革升级。智慧物流已经成为当今物流行业升级的一个重要方向。区块链和物联网已成为智慧快递物流中重要的技术手段。区块链技术作为一门新兴技术，因其分布式储存、去中心化、去信任化的特性受到广泛关注，其具有节省中介成本、信息防伪、任何节点交易顺利、信息不可篡改等特点，区块链技术的引入可以为物流中用户隐私泄露、过程不透明、赔偿机制不透明等基础问题提供技术手段。

AIoT 即人工智能加物联网，其将人工智能技术与物联网技术相融合，利用 GPS 卫星导航定位技术、RFID 技术、传感技术等多种技术，在快递运输过程中可实现在实时车辆定位、实时运输物品温湿度、烟雾监控，保证了快递运输的安全。

综上所述，基于 AIoT 和区块链技术的智慧快递运输管理系统对快递行业的发展以及提高快递物流企业管理效率十分重要。本作品采用 AIoT 技术以及区块链技术，实时监控车内状态，用户可以通过系统实时查看相关数据，保证了快递运输的安全；利用区块链技术实现信息的可追溯，通过系统可以查看快递包裹的信息，保证了整个运输过程的透明；采用遗传-蚁群融合算法实现多点路径规划，系统可以根据目的地生成多条路径，选择最优路径，提高了快递配送效率。最终构建整个快递运输管理系统，保障快递物流服务质量，促进智慧快递等战略规划发展目标更快更好地实现。

1.2 存在问题

根据目前快递物流运输行业的现状，目前存在的问题如下：

在途跟踪困难，运输过程安全性差。传统的快递运输过程中平台难以实时定位车辆位置，使管理者无法及时获取运输车辆位置，难以及时监测快递运输的各种状态。此外，驾驶员在运输过程中容易由于自身状态例如疲劳驾驶，容易导致快递运输安全事故的发生，不利于快递运输行业安全健康发展。

信息难以追溯，用户隐私容易泄露。当前我国物流行业特别是快递业务，在传统的技术系统中很难实现信息透明化、溯源化以及不可篡改，这导致快递包裹的信息难以追溯。此外，目前快递运输是通过包含用户明文信息的快递单来实现的，由于贴在包裹上的快递单暴露在外，使得不法分子很容易直接或间接地得到用户信息，从而造成用户隐私泄露，快递客户自身的信息安全很难得到保障，这势必会对物流快递行业的信息安全、自动化与智能化发展带来不利影响。

配送效率低下，物流路径缺乏规划。现有的物流配送系统运行效率低下，运输过程组织不合理、功能衔接关系不协调，选择的路径并未达到最优，货运车辆的交错运输让物流配送的时间性难以保障。这不仅带来较高的物流成本和低下的服务水平，还影响着以物流配送为基础的综合运输业和电子商务的发展，而且带来了交通堵滞、环境恶化等负面效应。

1.3 系统功能

(1) **基于 AIoT 的在途跟踪和安全保障功能。**系统通过 GPS 及时采集快递车的位置信息，实现车辆的实时在途跟踪。通过温度、湿度、烟雾传感器采集车内温湿度以及烟雾浓度，防止部分特殊物品因为环境因素造成损坏。系统实时监控司机的状态，通过人工智能技术识别司机疲劳状态，并提醒司机，降低因司机疲劳驾驶导致安全事故发生的概率。

(2) **基于区块链技术的信息追溯和隐私保护功能。**系统采用区块链技术，从供应链的源头开始，利用物联网、互联网的物品身份标识以及传感器等技术，可以将快递包裹运输过程所有的信息都实时地记录在区块链上，透明安全且不可篡改，实现了快递包裹运输全过程信息可追溯的功能。此外，区块链技术能够确保交易数据不被篡改，在此基础上我们设计了一个基于层次身份的加密算法，并利用该算法对快递用户信息进行分段加密，保障用户的隐私安全。

(3) **基于改进的遗传-蚁群融合算法的路径规划功能。**系统采用基于改进的遗传-蚁群融合算法的路径规划方法，该方法综合利用了遗传算法的全局随机搜索能力和蚁群算法的优化特性，在路径规划过程中能够快速收敛，避免陷入局部最优解，同时还能够获得较低的总运输成本。且此算法综合考虑了快递运输车辆的固定和行驶成本、时间惩罚成本以及货损成本等多种因素，具备较高的实际应用价值，提高了配送效率，降低了运输配送成本。

1.4 意义价值

如今国家正在持续推进智慧快递建设，快递行业逐渐向数字化、信息化迈进。解决运输过程中**实时跟踪困难、快递损坏率高、运输安全难保障**；快递系统中**快递包裹信息难以追溯、用户个人信息泄露**；快递运输过程中，**运输路径开销大，效率低**等问题的解决是降低快递企业配送成本、提高企业管理效率的关键。

综合上述问题，该项目设计并实现了**基于 AIoT 和区块链的智慧快递运输管理系统**，对传统的快递运输管理系统进行改进，让其在现实应用中有非常重要的实际价值。

本项目利用**人工智能、物联网和区块链**技术为快递行业提供信息化、智能化、可追溯的解决方案。实现了快递车内数据实时监控、对司机异常状态进行预警、

保证了快递运输的安全，降低了快递的损坏率。通过区块链技术实现了节点的分布式存储和智能合约，管理和解决各方之间的信任问题，提升整个快递运输系统的透明度和智能化水平，保障了用户信息的安全。通过遗传-蚁群融合算法实现路径规划，优化了配送路线，降低了能耗和碳排放量。

在实际应用中，本系统能够适应复杂多样的快递运输环境，应用覆盖范围广，具有普适性；友好的界面让快递运输系统管理化繁为简，能更有效地管理快递运输过程，具有高效性；能够帮助企业提高物流服务质量，降低运营成本，提高生产效率，具有实用性；有助于建设一个高效的快递运输体系，为实现智慧快递战略规划发展奠定基础，具有先进性。

第2章 设计思路与方案

2.1 系统总体设计

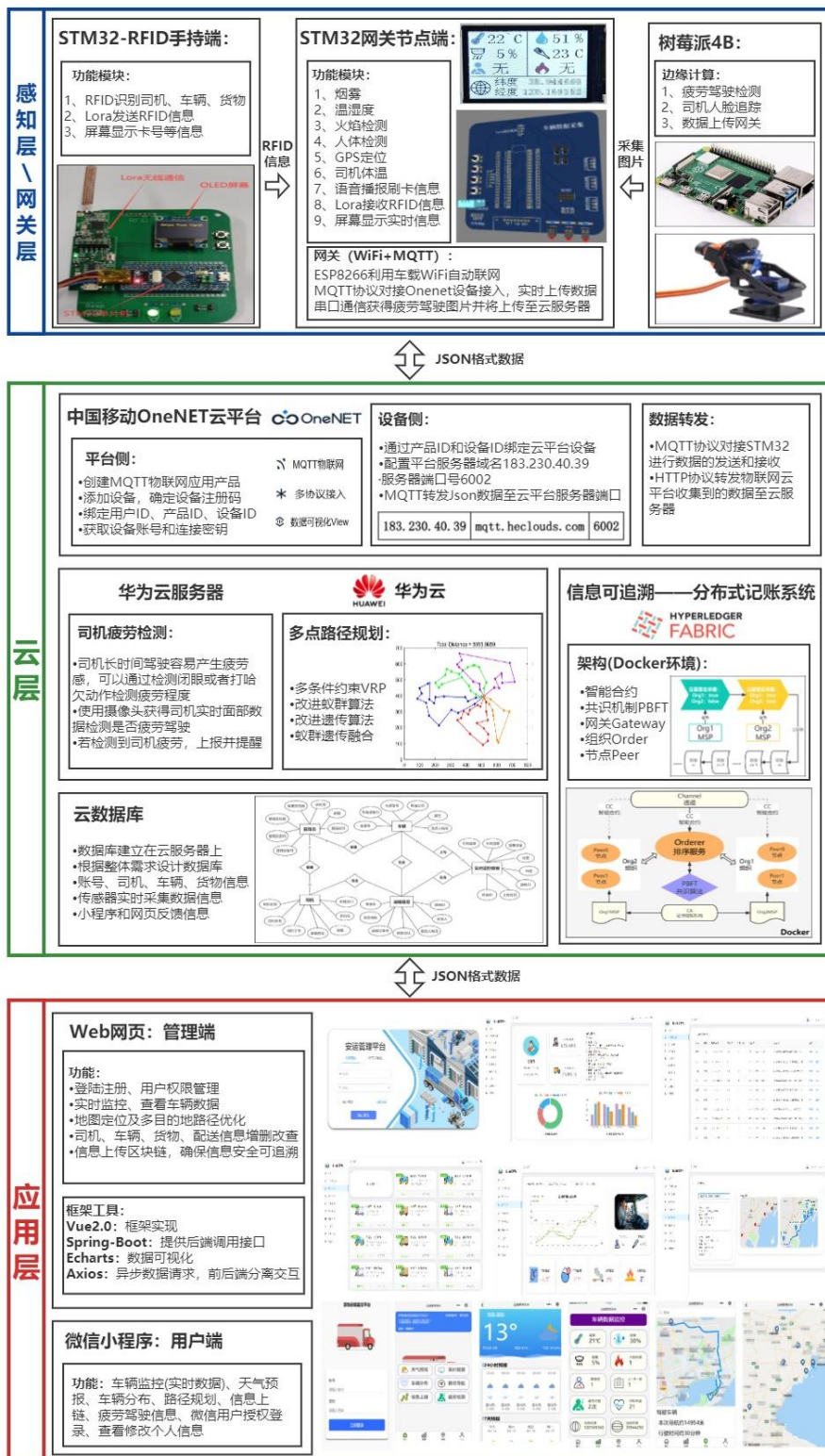


图 2-1 智慧快递运输管理系统结构图

本智慧快递运输管理系统主要由三层组成，包括感知层/网关层、云层和应用层。其中，感知层/网关层位于系统硬件部分，负责采集数据并上传至云平台 and 云服务器。云层包括云平台和云服务器，实现数据处理和分布式存储，同时搭建联盟链网络，确保数据可追溯。在应用层中，可以通过 Web 网页端来管理车辆、司机和配送信息，以及查看车辆实时数据。小程序则作为用户端，提供配送过程中实时数据查看和路径导航等功能。

2.2 感知层/网关层设计

本系统感知层/网关层的主要目标是采集车内环境参数，将多传感器数据进行融合，通过无线终端设备采集上传并实时存储，让用户可以通过远程访问服务器，获取车辆的实时定位和车厢内环境动态。

感知层主要由 STM32 数据采集部分和树莓派边缘计算部分组成。其中 STM32 单片机部分又分为 RFID 手持端和数据采集网关端。前者主要负责采集司机、车辆、货物的 RFID 编号并传输至网关节点进行人-车-货识别，后者的工作是通过传感器采集车内数据并通过 MQTT 协议上传至云平台 and 云服务器。

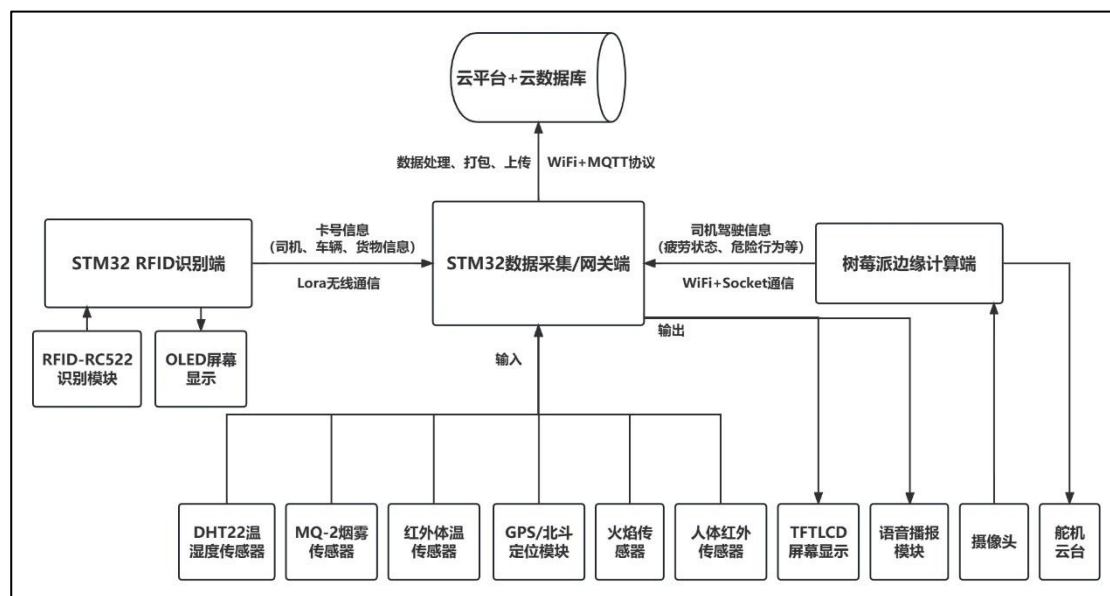


图 2-2 感知层/网关层系统架构图

1、多传感器融合参数监控。温湿度是快递运输车辆环境监测系统中非常重要的因素之一。为了更加真实地反映车厢内温度的变化特性，在快递运输车的前端、中部、尾部等位置布置温湿度传感器，并取其平均值后及时上传至云端。此外，在车内还应安装烟雾报警器、火焰探测器等设备来保障货车和货物的安全。

同时，还应该安置司机人体检测和体温监测模块，及时掌握司机身体状况，以保障货车的安全出行。



图 2-3 STM32 数据采集端实物图

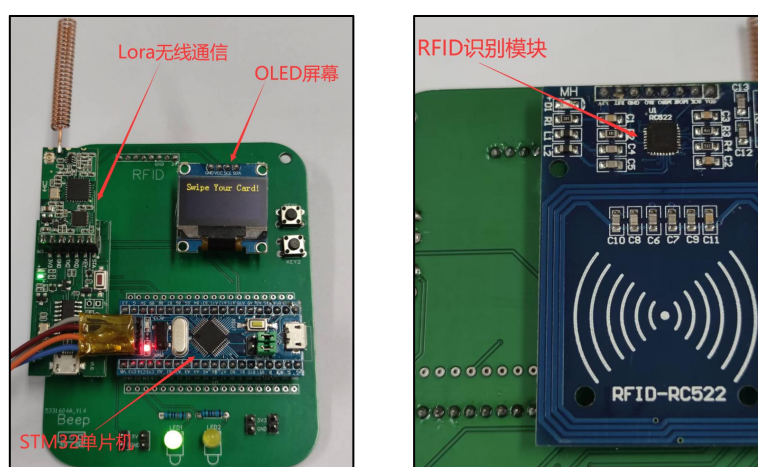


图 2-4 RFID 采集端实物图

2、车辆位置实时定位。本系统通过 GPS/北斗全球定位系统让用户实时查看车辆所在的地理位置，方便监控中心根据交通路况信息调度车辆的行驶路线，以降低运输成本。快递运输车辆实时定位系统与车载定位模块连接，将全球定位系统采集到的位置信息在 Web 页面上显示出来，车辆位置及其具体信息可以利用地图 API 进行实时显示，供用户查询、掌握。

3、车辆数据实时上传。我们在运输车辆内部安装移动 WiFi，方便车载设备对互联网的连接和访问，同时单片机自身作为网关，利用 MQTT 协议对数据收发进行管理。

RFID 读卡端读取卡号后将信息传送至 STM32 节点端，该节点自动采集温湿度、烟雾浓度等车内信息组成报文，每隔一段时间上传至 OneNET 云平台。同时也能接受来自树莓派边缘计算端发送的图片数据，每次当树莓派检测到司机有疲劳驾驶状态时，都会拍照并发送给单片机相应信息，单片机上传并作出相应的提醒。

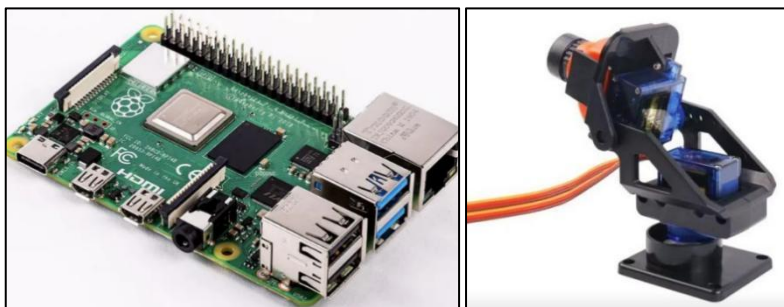


图 2-5 树莓派和二维云台实物图

2.3 云层设计

2.3.1 数据库设计

按照快递物流配送路径优化业务流程，本小节对系统的数据库部分进行设计，系统用到的数据表主要有：管理员信息表（admin）、司机信息表(driver)、车辆信息表(vehicle)、运输信息表(transport)、实时监控数据表（monitor）等。

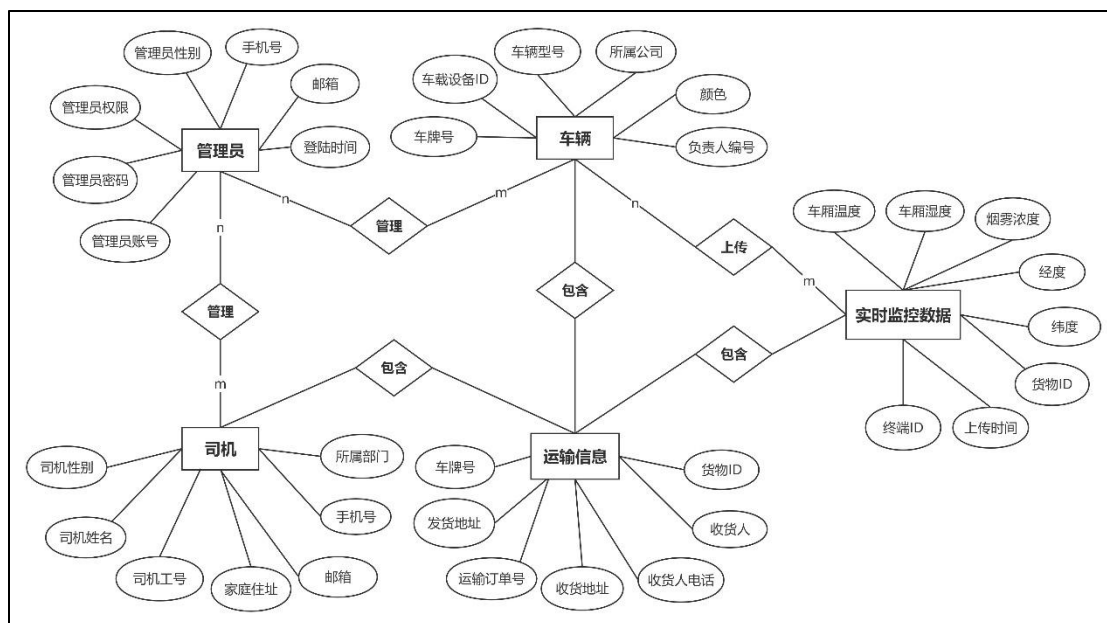


图 2-6 数据库 ER 图

(1) 管理员信息表

管理员表主要记录登陆信息、设置用户权限、保证系统安全，登陆远程管理系统可以查看相应的功能，包括登陆账号、登陆密码、性别、权限、手机号、邮箱号、用户登陆时间等。

表 2-1 管理员信息表

字段名	数据类型	长度(/bite)	约束	备注
AdminID	VarChar	16	主键	管理员账号
AdminPassword	VarChar	16		管理员密码
AdminSex	Bit	2		管理员性别
AdPhoneNum	VarChar	30		手机号
AdEmail	VarChar	50		邮箱
LogTime	SmallDateTime	20		登陆时间

(2) 司机信息表

司机信息表主要记录司机的主要信息，为应用端提供运输车负责人的信息，该表主要包括车辆负责人司机的工号、姓名、性别、所属部门、联系电话、邮箱、家庭住址等，当车厢温度出现异常发出报警，监控中心人员可以通过司机信息表得到车辆负责人联系方式与其进行联系，及时查看异常情况原因并进行及时抢修。

表 2-2 司机信息表

字段名	数据类型	长度(/bite)	约束	备注
DriverID	VarChar	16	主键	司机工号
DriverName	VarChar	16		司机姓名
DriverSex	Bit	2		司机性别
DrDepartment	VarChar	80		所属部门
DrPhoneNum	VarChar	30		手机号
DrEmail	VarChar	50		邮箱
DrAddress	VarChar	100		家庭住址

(3) 车辆信息表

车辆信息表用来接收车载终端发送的有关车辆的相关信息，主要包括车牌号、车载设备 ID、车辆型号、车辆来源公司、颜色、车辆负责人等。

表 2-3 车辆信息表

字段名	数据类型	长度(/bite)	约束	备注
VehNum	VarChar	20	主键	车牌号
VehSBID	VarChar	20		车载设备 ID
VehStyle	VarChar	20		车辆型号
VehSorce	VarChar	40		所属公司
VehColor	VarChar	10		颜色
VehPersonNum	Int	40	外键	负责人编号

(4) 运输信息表

运输信息表主要是快递运输过程中的订单，主要包括运输订单号、发货地址、运输车牌号、货物 ID、收货人、收货人联系电话、收货地址等。

表 2-4 运输信息表

字段名	数据类型	长度(/bite)	约束	备注
OrderNum	VarChar	20	主键	运输订单号
SourAddress	VarChar	100		发货地址
VehNum	VarChar	20	外键	车牌号
GoodID	VarChar	20	外键	货物 ID
ReceName	VarChar	20		收货人
RecePhone	VarChar	30		收货人电话
ReceAddress	VarChar	100		收货地址

(5) 实时监控数据表

实时监控数据管理表主要包括快递运输车终端设备 ID、测试温度、经度、纬度、RFID 采集的货物 ID、数据生成时间等，通过 ID 可查询到对应运输车、司机及货物的相关信息。

表 2-5 实时监控数据表

字段名	数据类型	长度(/bite)	约束	备注
VehSBID	VarChar	20	主键	终端 ID
Temp	VarChar	10		车厢温度
Humi	VarChar	10		车厢湿度
Smog	VarChar	10		烟雾浓度
Longitude	Decimal	18		经度
Latitude	Decimal	18		纬度
GoodID	VarChar	20	外键	货物 ID
UpTime	SmallDateTime	20		上传时间

2.3.2 路径规划算法设计

带有时间窗的车辆路径问题（VRPTW）是本项目力争解决的重要问题之一，也是现在物流行业和快递配送过程中最常出现的问题。它是指一定数量的客户，各自有不同数量的货物需求，配送中心向客户提供货物，由一个车队负责分送货物，组织适当的行车路线，目标是使得客户的需求得到满足，并能在一定的约束下，达到诸如路程最短、成本最小、耗费时间最少等目的，并**规划最优的路径**以满足用户需求。

为了解决这个问题，本系统采用了一种基于改进的遗传-蚁群算法的路径规划方法，该方法结合了遗传算法全局随机搜索能力和蚁群算法的优化特性，可以快速收敛，避免出现局部最优情况，同时还能够获得较低的总运输成本。

1、多条件约束内容

根据调研分析，本系统考虑了五大约束条件构成项目的多条件约束内容，分别是**车辆配送固定成本、车辆配送行驶成本、时间惩罚成本、货损成本和碳排放成本**，并将这五个约束条件应用到路径规划中，如图 2-7 所示：

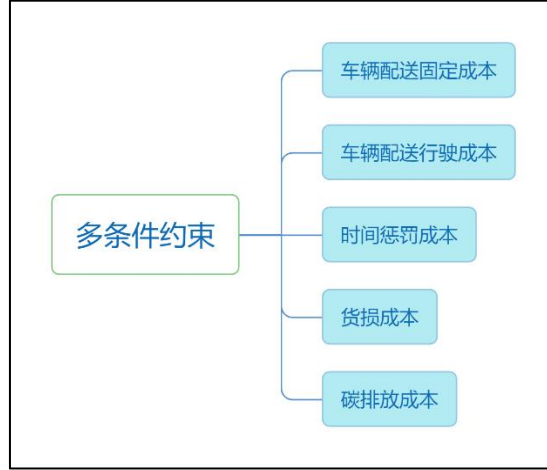


图 2-7 多条件约束内容

(1) 车辆配送固定成本

此成本包括货车自身损耗以及司机薪资等。假设配送中心的可用配送车辆总数为 v ，每辆车的固定成本为 C ，则在本次配送服务中总的固定成本为：

$$f_1 = C \sum_{k=1}^v \text{sign}(n_k) \quad (2-1)$$

$$\text{sign}(n_k) = \begin{cases} 1, & n_k \geq 1 \\ 0, & \text{otherwise} \end{cases} \quad (2-2)$$

通过上述公式可以判断出车辆 k 是否参加了配送任务。

(2) 车辆配送行驶成本

这部分成本主要是由于车辆的行驶距离增加而产生的，例如油耗，也是总运输成本中占比最大的一部分，与配送车辆实际的行驶距离成正比，则在运输过程中配送车辆的总行驶成本为：

$$f_2 = \xi \sum_{k=1}^v \sum_{i=1}^n \sum_{j=1}^n x_{ij}^k d_{ij} \quad (2-3)$$

其中 d_{ij} 表示客户节点 i 和客户节点 j 之间的距离，已知节点 i 与 j 在城市中的相对坐标分别为 (x_i, y_i) 和 (x_j, y_j) ，可调用地图 API 自动生成最优距离路线，求得 d_{ij} 。

(3) 时间惩罚成本

这部分所产生的成本主要是由于配送车辆在运输过程中未能按照客户所要求的时间限定范围来配送货品而产生的惩罚成本, 具体的惩罚金额由客户和配送中心或物流企业在订单生成时达成共识。 $p(t_i^k)$ 表示在 t 时刻由车辆 k 将货物配送到客户点 i 而产生的惩罚成本, 如公式所示:

$$p(t_i^k) = \begin{cases} m & t_i^k < a_i \\ \frac{m}{b_i - a_i} (b_i - t_i^k) & a_i \leq t_i^k \leq b_i \\ 0 & b_i \leq t_i^k \leq c_i \\ \frac{m}{d_i - c_i} (t_i^k - c_i) & c_i \leq t_i^k \leq d_i \\ m & t_i^k > d_i \end{cases} \quad (2-4)$$

因此本次配送的总惩罚成本为:

$$f_3 = p(t) = \sum_{i=1}^n p(t_i^k) \quad (2-5)$$

(4) 货损成本

快递配送运输的产品也会受到空气的温度、湿度、含氧量等因素的影响, 形成一定程度的损耗, 当损耗达到某一阈值时会产生相应的货损。对于产品的货损, 引入质量损耗函数 $D(t) = D_0 e^{-\hat{\alpha} t}$, 其中 $D(t)$ 表示在 t 时刻的产品质量, D_0 表示在配送中心出发时的产品质量, t 代表产品所经历的运输时间, $\hat{\alpha}$ 表示产品的腐败率。

整个配送过程产生的货损成本 f_4 可表示为:

$$f_4 = P_4 \left(\sum_{k=1}^K \sum_{i=0}^N \sum_{j=1}^N x_{ijk} Q_i (1 - e^{-\varepsilon_i t_{ij}}) \right) + P_4 \left(\sum_{k=1}^K \sum_{i=0}^N \sum_{j=1}^N y_{ik} (Q_i - q_i) (1 - e^{-\varepsilon_i T_{ij}}) \right) \quad (2-6)$$

(5) 碳排放成本

碳排放量与油耗量成一定线性关系, 碳排放量是燃料消耗量和二氧化碳排放系数之积。整个配送过程产生的碳排放成本 f_5 可表示为:

$$f_5 = P_3 C_e x_{ijk} \left[\sum_{k=1}^K \sum_{i=0}^N \sum_{j=1}^N F_1 + \sum_{k=1}^K \sum_{i=0}^N \sum_{j=1}^N (F_2 + F_3) \right] \quad (2-7)$$

由此得到带时间窗的车辆路径优化模型的优化目标函数为：

$$F = \min(f_1 + f_2 + f_3 + f_4 + f_5) \quad (2-8)$$

2、改进的遗传-蚁群融合算法

我们提出了一种改进的遗传-蚁群融合路径优化算法，通过融合两种算法的优点，充分利用改进的遗传算法的随机搜索和全局收敛特性，弥补了蚁群优化搜索速度慢和易陷入局部最优的缺点。

(1) 改进的遗传算法根据适应度值的大小来动态调节 P_c 和 P_m ，从而实现算法的全局搜索性和随机性。即将问题的解经过选择、变异和交叉等一系列的操作，编码成满足模型提出的约束条件的染色体，充分的利用了单一遗传算法高效并行和全局搜索的优点。

$$P_c = \begin{cases} P_{c \max} - \frac{(P_{c \max} - P_{c \min})(f_{\text{avg}} - f')}{(f_{\text{avg}} - f_{\min})} & f' \leq f_{\text{avg}} \\ P_{c \max} & f' > f_{\text{avg}} \end{cases} \quad (2-9)$$

$$P_m = \begin{cases} P_{m \max} - \frac{(P_{m \max} - P_{m \min})(f_{\text{avg}} - f)}{(f_{\text{avg}} - f_{\min})} & f \leq f_{\text{avg}} \\ P_{m \max} & f > f_{\text{avg}} \end{cases}$$

其中, f_{avg} 和 f_{\min} 分别指种群中平均适应度值和最小适应度值, f' 为进行交叉操作的双亲中较大的适应度值, f 为进行变异的亲代个体的适应度值, 其中 $P_{c \max} > P_{c \min} \in (0,1)$, $P_{m \max} > P_{m \min} \in (0,1)$ 。

(2) 将遗传算法的求解的最优解作为蚁群优化的初始化信息素浓度 $p_{ij}^k(t)$ 。

其中, ε_1 和 ε_2 分别为信息量权重系数和时间窗权重系数, 取值均为 $(0,1)$, 且满足 $\varepsilon_1 + \varepsilon_2 = 1$, 通过上面提到的时间窗函数可知 $[b_j, c_j]$ 为客户要求的时间窗范围。

具体算法设计流程如下所示：

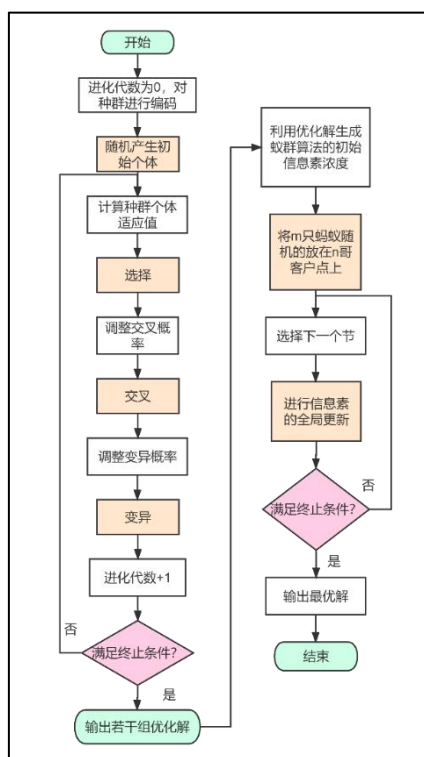


图 2-8 改进遗传-蚁群融合算法流程图

2.3.3 联盟链设计

为确保数据安全存储，本系统采用基于 Docker 的 Fabric 开源联盟链技术进行信息安全设计，也就是分布式账本记录方式，将不同的节点构建成不同的组织，基于组织之间的协作来实现数据的同步和备份。该技术可单服务器开发、多服务器部署，确保数据安全稳定。联盟链具有多个参与方，所有交易必须得到所有参与方认可，提高了数据安全性、可靠性，减少维护成本，有利于企业发展。

考虑到实际情况资金和资源的限制以及 Docker 的可拓展性和实践的便利共用，我们决定以 Docker 容器作为基础架构，选择开源的 Fabric 镜像，在其中构建了两个组织和一个排序节点，并将每个组织下添加了两个对等节点。未来，我们还将根据需求，增加更多组织节点和对等节点，并部署它们到不同的服务器上，以提高系统的容错性和可拓展性。

1、网络结构设计：

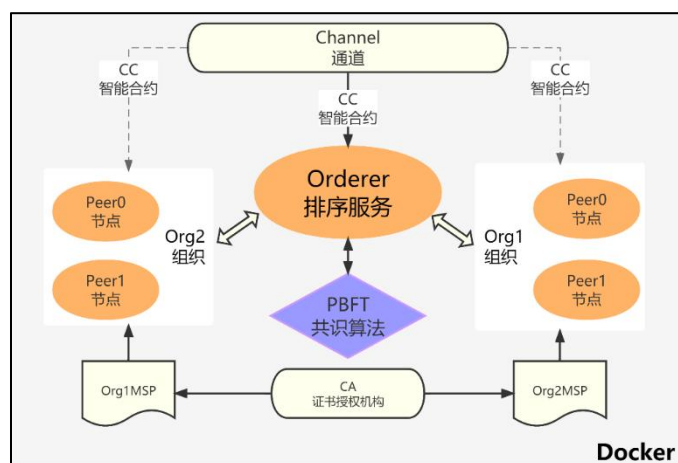


图 2-9 Fabric 网络结构设计

为了使联盟链网络能够顺畅地运行，我们在所有组织和节点之间创建了一个统一通道，并将 recordplans 智能合约部署到该通道上。同时，Fabric 提供了可插拔的共识方案，方便我们对共识算法进行定制和开发。

在这里，我们采用 PBFT 算法作为共识机制，以提高少数节点中恶意节点的容错性，并适度扩展联盟链的规模。在实际的多服务器部署中，我们将根据具体情况来选择合适的共识机制、通道建立和组织划分等参数，以最大限度地实现联盟链的安全性和可拓展性。

2、背书设计：

针对网络中节点较少的情况，本系统采用全部组织节点背书策略，要求所有组织节点都要对交易进行签名认证，只有在得到全部组织节点的认证后，交易才能上块。

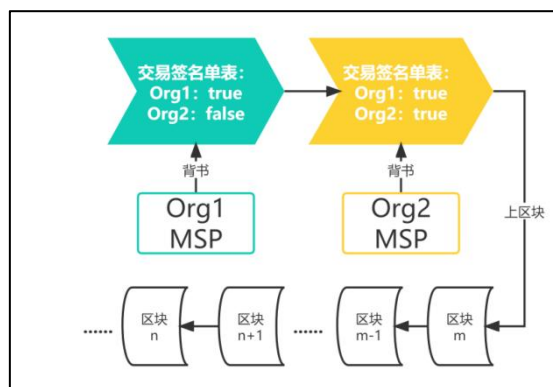


图 2-10 交易上块设计图

3、智能合约与 API 设计：

内部智能合约主要实现增删改查四大功能，还需要编写一个初始化函数来建立创世区块。

首先定义结构体变量 Plan，来代表区块中每条记录的具体内容：

```
type Plan struct {
    Record      string `json:"Record "` //记录号
    Driver      string `json:"Driver "` //司机姓名
    Car         string `json:"Car "` //货车编号
    ArrivalTime string `json:"ArrivalTime"` //到达时间
    Phone       string `json:"Phone"` //联系号码
    Address     string `json:"Address"` //地址信息
}
```

接下来是函数设计：

表 2-6 Fabric 智能合约函数列表

函数名称	形参说明	返回内容
initLedger	无	error: 报错信息
CreatePlan	Plan: 记录内容	error: 报错信息
DeletePlan	Record string: 记录号	error: 报错信息
UpdatePlan	Plan: 记录内容	error: 报错信息
ReadPlan	Record string: 记录号	Plan: 记录内容

Fabric 提供了一个用于简化使用的网关接口，使得用户可以使用 typescript 编写应用，并利用 express 库函数暴露应用端口。通过这种方式，用户可以构建全新的 API 端口，与后端系统进行交互。为了确保后端系统与数据存储管理系统之间的分离以及提高安全性，我们只提供记录和查询功能，而删除和修改功能只能由 Fabric 网络管理员进行操作，不再提供简易 API 应用。

对于外部 API，出于安全考虑，我们只暴露读取和记录两个功能，并建立 gRPC 连接与联盟链网络进行交互。具体设计如下：

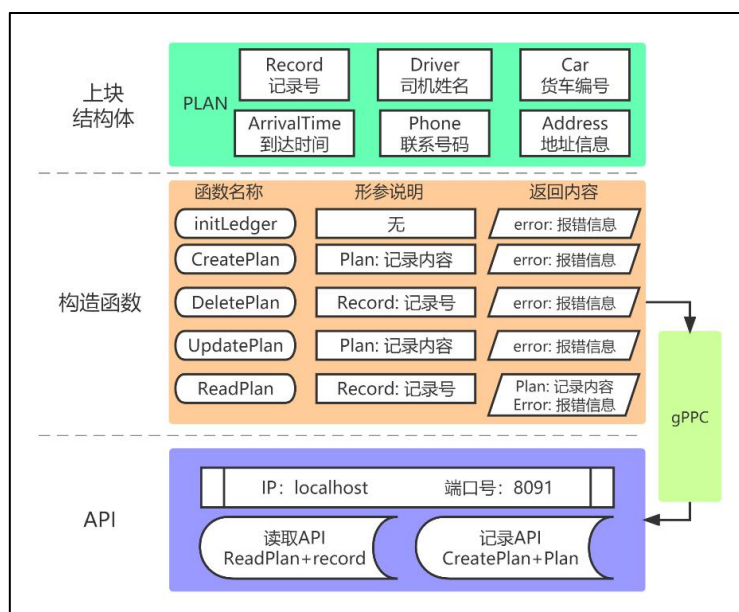


图 2-11 智能合约与 API 结构

2.4 应用层设计

应用层主要分为两大部分，分别是网页管理员端和小程序用户端。前者网页端主要面向的是系统的管理员和维护人员，针对全部司机和车辆的信息进行管理，统筹安排司机的多目的地路径优化问题。后者小程序端面向的是司机用户，可以实时查看本车的各类参数和车辆状况，根据路径规划确定配送的最优路线，极大提高司机的配送效率。

2.4.1 网页端

网页端主要基于 Vue2.0 和 SpringBoot 前后端分离搭建而成，使用 php 实现网页与云数据库的连接与通信，前端利用 ajax、html、Jquery 实现交互动态网页。实现了账号关系、权限分发、司机车辆信息的增删改查、配送申请、车辆监控查看实时数据、数据可视化、本系统货车分布图、多目的地路径规划、配送信息写入联盟链、反馈处理等功能。

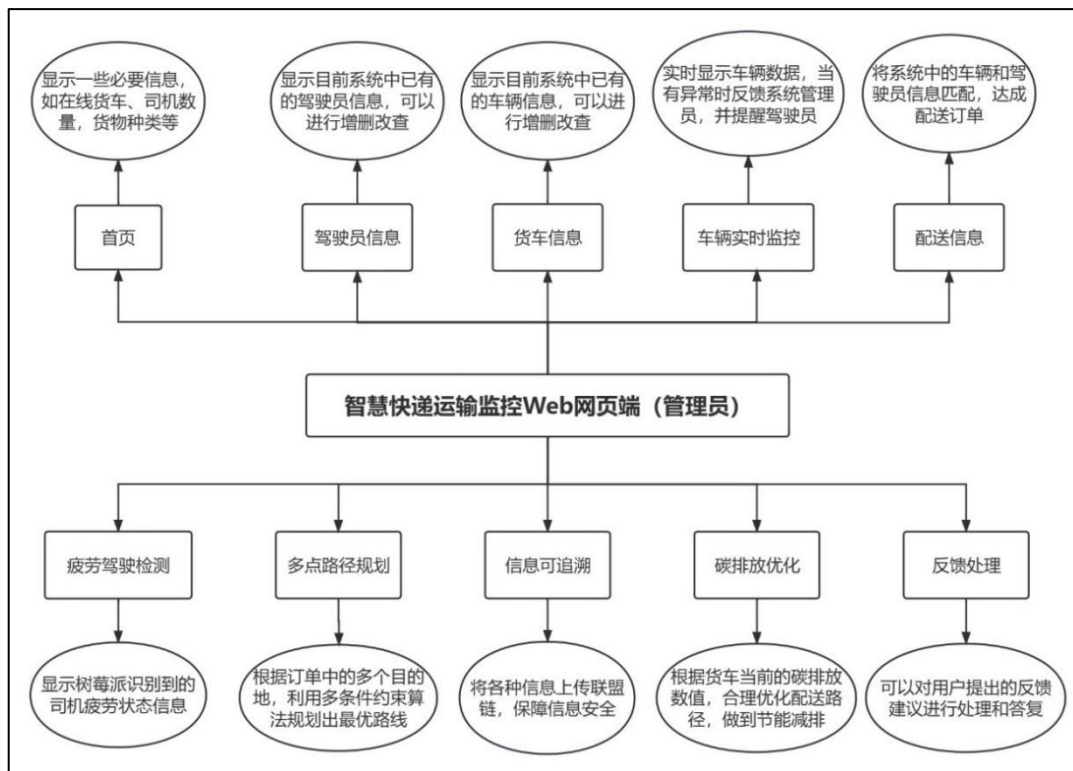


图 2-12 Web 网页功能流程图

网页端主要面向的是企业的管理员，用来显示货车运输的相关信息，包括在线司机数、在线货车数、运输物品相关信息、货车分布图等。网页端还可以实时显示车内温湿度、烟雾浓度、火焰检测情况、驾驶员的状态和体温等情况。它还提供给管理员多点路径规划的功能以及高效追溯货物信息的功能。

网页端通过可视化界面展现从终端设备收集的车厢内多传感器融合数据和司机状态信息，同时利用 GPS 定位和地图 API 进行位置展示，方便管理员实时监测车辆状态。此外，网页端还提供路径优化方案和碳排放变化的可视化展示，同时与联盟链网络交互，为相关信息提供可追溯功能，确保信息安全。

2.4.2 小程序端

小程序端为用户端，主要面向的是广大货运司机用户，为司机提供天气预报、实时车辆数据监控、车辆分布图、路径规划导航、疲劳驾驶检测、货运信息写入联盟链、反馈建议等功能。

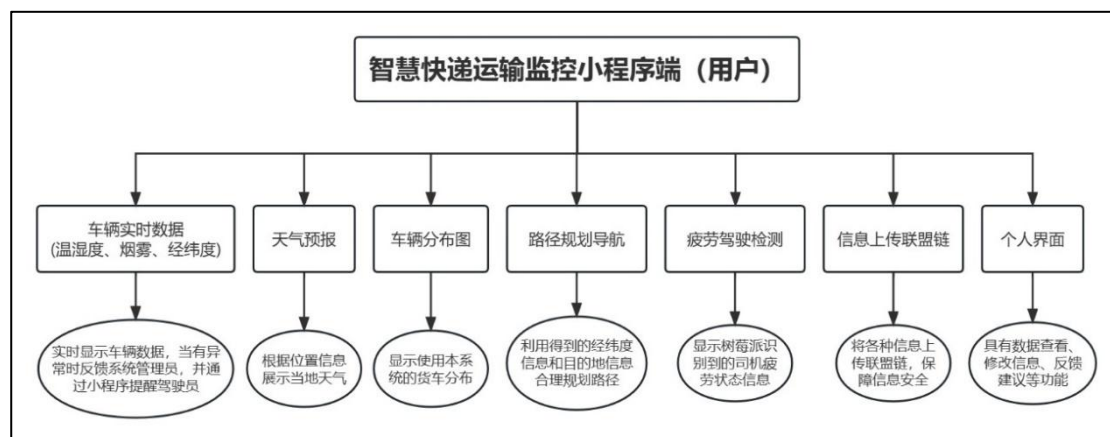


图 2-13 小程序功能流程图

车辆硬件设备会收集车厢内的多传感器融合数据和 GPS 信息，上传到云端进行处理，并将结果显示在小程序页面上供驾驶员查看。如果发生异常情况，如检测到火焰，小程序会及时向管理人员发送消息，并提供驾驶员详细位置信息。除此之外，我们还将安装摄像头，用于监测驾驶员的驾驶状态，以便在驾驶员出现疲劳状态时提醒其注意安全。

利用路径规划算法匹配出的最优配送路径，会自动分配到小程序司机的导航设置。其中在路线规划、货车分布等功能上使用了腾讯地图服务等接口，也调用了云平台数据库接口和云服务器的接口，用来接收和发送数据报文，实现用户信息交互。

小程序端也有数据上传联盟链的备用功能，我们可以将配送时间、车辆信息、司机信息、货物信息、始末地点信息等打包后上传至联盟链，实现信息可追溯功能，防止后端程序故障而导致的物流追踪瘫痪。

第 3 章 方案实现

3.1 开发环境要求

编译环境：

- Go 版本：Go 1.15.7
- Fabric 版本：2.4.9
- Docker 版本：Docker Desktop Community 20.10.6

开发环境：

- VS Code 版本：1.56.2
- Node.js 版本：v14.17.0
- Yarn 版本：1.22.10
- TypeScript 版本：4.3.2
- WSL 版本：WSL2

3.2 感知层/网关层实现

3.2.1 数据采集

数据采集功能基于感知层进行实现，其包括 STM32 单片机和树莓派边缘计算两部分。STM32 单片机负责采集不同传感器的数据，并将其上传至云平台 and 云服务器。树莓派边缘计算则用于检测司机是否疲劳驾驶，并拍摄相应照片上传提醒。

为了提高硬件部分的集成度，我们绘制了电路板 PCB 来将 STM32 单片机和其他各类传感器整合在一起，提供更多的硬件扩展接口，方便进行重烧录和增添新传感器。网关端主要通过使用无线传感信息接收装置 Lora 等，接收车厢信息采集控制系统传递的车厢环境信息，车头控制中心可以从里程表获取车辆运行信息，用于检测车辆是否已停止。数据采集实物图如图 3-1 所示：

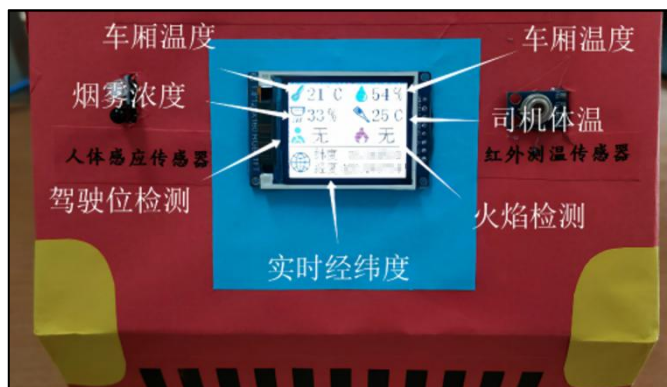


图 3-1 数据采集实物图

STM32 数据采集部分：

由两部分组成，RFID 终端用到射频识别模块读取司机、车辆、货物的 UID 卡号，并通过 Lora 通信将读取到的卡号传输给网关节点。信息采集网关终端负责采集车内的各类参数和信息，上传数据至云平台和服务端。主要功能和实现如下：

- DHT22 温湿度传感器读取车舱内的环境温湿度；
- MQ-2 烟雾传感器 ADC 转换处理后读取空气中的烟雾浓度；
- 火焰传感器利用自带的模数转换检测车舱内是否发生火焰；
- 人体感应传感器和 MLX90614 红外温度传感器检测司机是否在驾驶舱以及司机的实时温度；
- 通过 GPS/北斗模块采集经纬度信息并使用 NMEA 协议将串口收到的报文解析出正确的经纬度数据；
- 接收来自 RFID 端和树莓派端发送过来的数据，这些数据都能够在终端屏幕上实时显示，将采集到的数据处理打包后通过 ESP8266 模块使用 MQTT 协议上传至云平台 and 云服务器。

树莓派边缘计算部分：

首先使用树莓派 4B 进行数据采集，配合使用二维云台人脸追踪，该装置可以根据驾驶人员的头部姿态和位置，自动调整摄像头的角度和方向，并保持视野内的人脸跟踪，提高驾驶的舒适性和体验，实现对驾驶人员状态的自动监控与报警。疲劳驾驶检测的主要依据是闭眼的程度、低头程度和是否打哈欠，如果树莓派检测到司机有以上行为，会自动拍照并上传服务器以及反馈给单片机提醒司机。

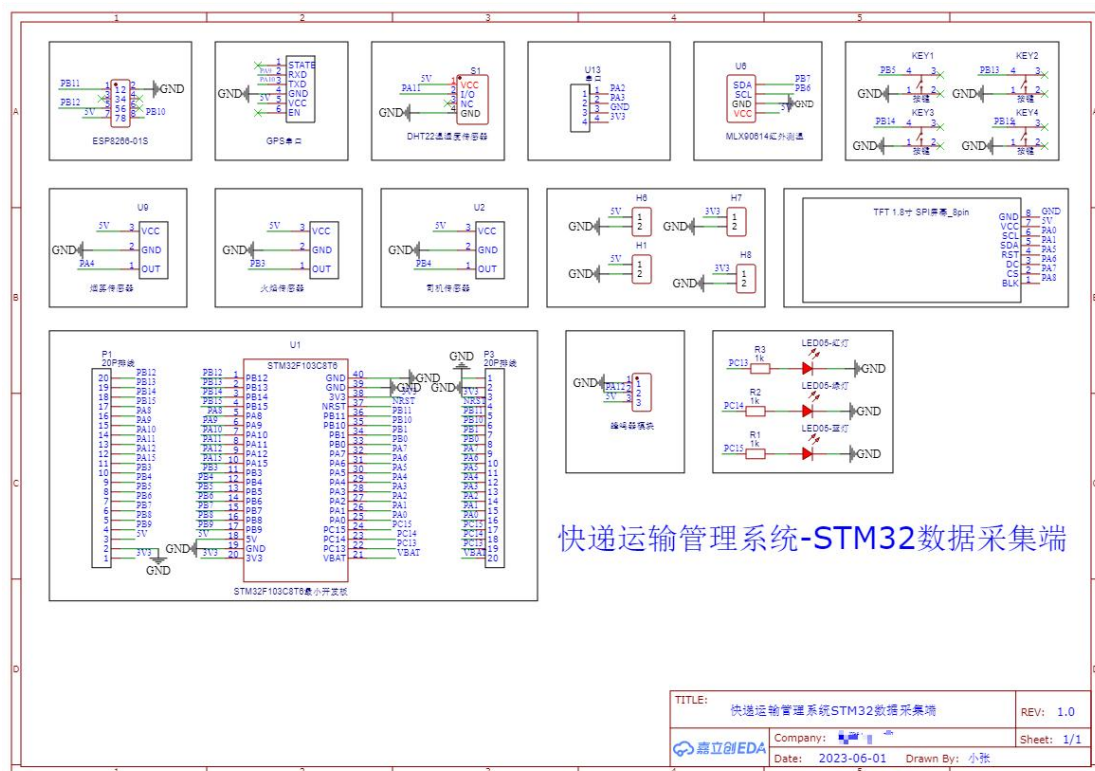


图 3-2 STM32 数据采集端电路图

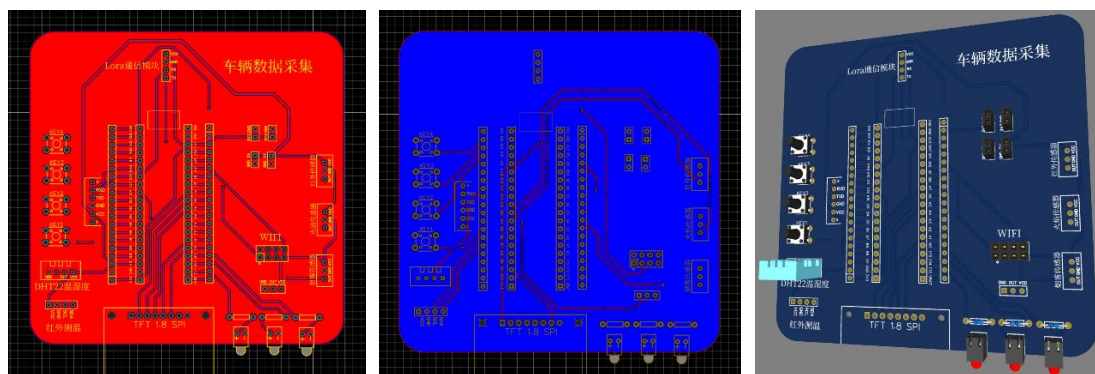


图 3-3 STM32 数据采集端 PCB 电路板预览图

上图为本项目中设计的电路图，其中包含了信息采集端使用到的所有传感器接口和预留的扩展接口。PCB 电路板能将多个电路接在一起，有效地减少空间占用，使原本繁杂的电路变得紧凑，同时也能减少杜邦线飞线连接，提高系统的稳定性。

3.2.2 多传感器数据融合

使用单一传感器无法克服由于硬件因素而产生的误差，上传的数据精准度可能会受到影响。同时，如果单一传感器损坏或者受到长时间的干扰也会影响传感

器对温度信息的采集，难以满足系统鲁棒性要求。同时也无法满足人们对数据信息丰富度、实时性、准确可靠性等方面的要求。

如果采取部署多传感器的方式进行温度数据的采集，每一个传感器独立采集一定区域的温度数据，所测得的数据具有互补性，可以一定程度上克服对于单一传感器硬件所造成的误差，或者受到一定时间干扰而形成的数据误差。多传感器采集的温度数据经过一些列处理后，由温度数据融合模型最终得到一个可信度和精准度较高的温度数据。

数据融合基于对多源数据的采集与处理，根据多个传感器在不同时间、空间、来源采集的数据，利用计算机技术按照时间序列获得经过分析的数据，对这些数据按一定的规则进行分析、聚类等操作，利用融合算法将多源、异构数据进行处理，产生新规则或有效信息，以备决策分析。

采取部署多传感器的方式进行温度数据的采集，每一个传感器独立采集一定区域的温度数据，所测得的数据具有互补性，可以一定程度上克服对于单一传感器硬件所造成的误差，或者受到一定时间干扰而形成的数据误差。在汇总所有子节点上传的信息后，利用 K-Means 聚类法剔除因长时间收到干扰或损坏的子节点上传的温度数据。多传感器采集的温度数据经过一些列处理后，由温度数据融合模型最终得到一个可信度和精准度较高的温度数据。

$$\bar{T}_k = \frac{k-1}{k} \bar{T}_{k-1} + \frac{1}{k} T_k \quad (3-1)$$

$$C_{i+1}(j) = \frac{1}{N_j} \sum_{T \in P_i(j)} T \quad (3-2)$$

因此本系统采用**自适应加权算法**对经筛选过的数据进行融合，利用单个温度传感器的局部估计值 T_i ，与最终温度融合值 T 的方差，在总均方误差最小这一最优条件下，自适应地调节各传感器的权重，寻找各个传感器的最优加权值，最大化的减少误差较大的温度传感器对最终温度融合值的影响。

$$\min S^2 = \min E \left[\sum_{i=1}^n \omega_i^2 (T - T_i)^2 + 2 \sum_{\substack{i=1, j=1 \\ i \neq j}}^n \omega_i \omega_j (T - T_i)(T - T_j) \right] \quad (3-3)$$

$$\tilde{T} = \sum_{i=1}^n \omega_i T_i \quad (3-4)$$

3.2.3 数据处理

①数据解析：信息发送至云端之后，需要对数据包进行解析。根据制定的相应通讯协议，对温湿度等数据进行解析。接收到的经纬度信息需要通过腾讯地图 SDK 和 API 解析后才能在网页上动态显示出来。

②数据存储：车载远程终端获取的数据上传并存储在云数据库之中，方便管理人员查阅分析。将运输过程中货物的环境信息发送至云平台，平台与运输端数据同步就此实现。车载终端传输数据被系统接收后会在数据库内保存，满足数据库备份与存储需求，方便前端网页和微信小程序实时显示。

3.2.4 网关数据上传

网络层是实现智慧快递物流智能化的重要结构，主要目的是将感知层采集到的数据通过有线或无线网络等信息通讯技术，快速可靠的传输至各需求方和不同的组织架构层去，起到承上启下的作用。由于快递物流体系的复杂性，由感知层采集的信息就必须通过网络层进行数据传输。如何将采集到的数据快速安全的传输至应用层成为网络层需要考虑的问题。

本系统使用 ESP-01S 作为 WiFi 通信模块，在程序里配置好车载 WiFi 的名称和密码，开机后自动连接。使用中国移动的 OneNET 云平台作为系统数据暂存和转发的媒介，前后端调用云平台的 API 接口读取多传感器数据和司机状态。

单片机作为网关，集成度高且使用方便。云平台下发的数据和传感器读取到的设备数据都会进入指定的消息队列。对于下发的 Json 命令，网关对其解析并执行相关操作。对于上传的数据，网关先对数据进行 Json 格式的拼装，再通过 MQTT 协议上传至云平台。节省了向外通信所需的能量。

3.3 云层实现

3.3.1 疲劳驾驶检测

本系统中设计了一种基于人脸检测与眼球跟踪的疲劳驾驶检测方法，该方法首先采用 Adaboost 算法对人脸位置进行初步判断，然后根据人脸的先验知识分割出大致的人眼区域。对识别出的大致人眼位置进行图像增强处理。根据人为设定好人眼矩形框的大小，覆盖上一步中确定的左右眼的中心位置，根据矩形框内的像素特征来判断眼睛的睁开闭合状态。

Adaboost 是一种迭代算法，其核心思想是针对同一个训练集训练不同的弱分类器，然后把这些弱分类器集合起来，构成一个更强的最终分类器。该算法本身是通过改变数据分布来实现的，它根据每次训练集之中每个样本的分类是否正确，以及上次的总体分类的准确率，来确定每个样本的权值。将修改过权值的新数据集送给下层分类器进行训练，最后将每次得到的分类器最后融合起来，作为最后的决策分类器。

首先，初始化训练数据的权值分布。每个训练样本初始都被赋予相同的权值： $1/N$,

$$D_1 = (w_{11}, w_{12}, \dots, w_{1i}, \dots, w_{1N}), \quad w_{1i} = 1/N, \quad i = 1, 2, \dots, N \quad (3-5)$$

进行多轮迭代，用 $m = 1, 2, \dots, M$ 表示迭代的轮次，使用具有权值分布 D_m 的训练数据集学习，得到基本分类器（选取让误差率最低的阈值来设计基本分类器）。

$$G_m(x): X \rightarrow \{-1, +1\} \quad (3-6)$$

计算 $G_m(x)$ 在训练数据集上的分类误差率，

$$e_m = P(G_m(x_i) \neq y_i) = \sum_{i=1}^N w_{mi} I(G_m(x_i) \neq y_i) \quad (3-7)$$

由上述式子可知， $G_m(x)$ 在训练数据集上的误差率 e_m 就是被 $G_m(x)$ 误分类样本的权值之和，计算 $G_m(x)$ 的系数， α_m 表示 $G_m(x)$ 在最终分类器中的重要程度，也就是基本分类器在最终分类器中所占的权重。最后组合各个弱分类器，

$$f(x) = \sum_{m=1}^M \alpha_m G_m(x) \quad (3-8)$$

从而得到最终的分类器，如下

$$G(x) = \text{sign}(f(x)) = \text{sign}\left(\sum_{m=1}^M \alpha_m G_m(x)\right) \quad (3-9)$$

3.3.2 改进遗传-蚁群融合算法实现多点路径规划

我们搜寻并整理了某物流公司当日的物流配送需求，并根据我们的多条件约束，进行配送需求的信息整理。基于改进遗传-蚁群融合算法实现多点路径规划可以应用于快递配送行业中，以优化物流配送路径以及减少运营成本。该算法结合了遗传算法和蚁群算法的优点，通过对传输路径进行优化，达到快速、高效和减少运营成本的目的。

具体而言，该算法会先经过一系列的初始化和参数设定，如路线点、车辆容量、路径距离、遗传算法参数和蚁群算法参数等。然后，根据问题的特征和不同影响因素，设计适宜的遗传算法和蚁群算法策略，利用遗传算法的优化思想来进行求解，同时利用蚁群算法中蚂蚁的概念实现路径规划。

例如，在快递配送过程中，初始设定运输区域，并设定每个区域的配送点。再根据车辆的容量、距离和最短时间等影响因素，设计遗传算法和蚁群算法的优化方法。在遗传算法中，针对路径的交叉和变异进行优化和策略调节，以获取符合期望的路径结果。在蚁群算法中，引入蚁群的移动选择和概率密度函数等算法，实现在路径选择与方向分配中的优化。

最终，该算法可以生成多条最优路径，并依据不同的优化目标来选择最佳路径，包括运输时间、路程和配送成本等因素。在实际的快递业务中，这种路径规

划算法可以应用于订单分配、配送中心的路径规划以及大规模配送网络优化，从而提高配送效率、降低运营成本，为企业提供更好的竞争力。

1、数据准备

我们搜寻并整理了山东省某市某物流公司当日的物流配送需求，并根据我们的多条件约束，进行配送需求的信息整理，整理表格如表所示：

表 3-1 客户需求表单

客户编号	货物类型	X 坐标(km)	Y 坐标(km)	时间窗
1	花卉植物	150	40	[03.16 13:24,03.16 15:30]
2	花卉植物	20	93	[03.16 10:14,03.16 12:34]
...
75	化学药品	64	74	[03.16 09:16,03.16 10:26]
76	冷藏食品	172	220	[03.16 19:19,03.16 20:50]

2、代码实现

具体改进遗传-蚁群融合算法的实现流程如图 3-4 所示：

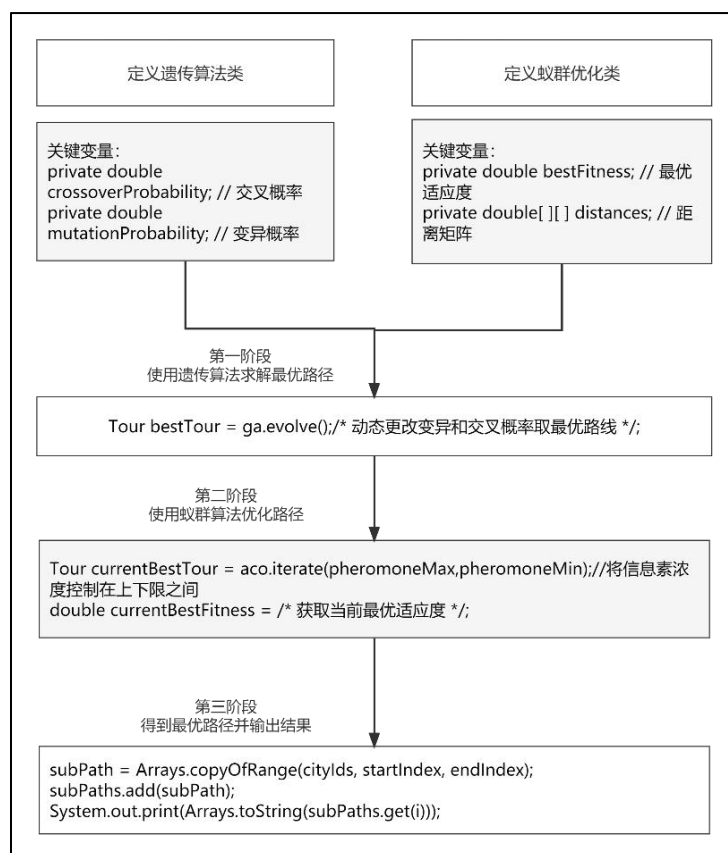


图 3-4 算法实现流程

定义遗传算法类

```
public class GeneticAlgorithm {  
    private double crossoverProbability; // 交叉概率  
    private double mutationProbability; // 变异概率  
    private int populationSize; // 种群大小  
    private int currentGeneration; // 当前代数  
    private int maxGenerations; // 最大代数  
    private int elitismCount; // 精英数量  
    private int tournamentSize; // 锦标赛选择的参与者数量  
    private boolean maximize; // 是否进行最大化  
    private Tour[] population; // 种群
```

定义蚁群优化类

```
private double alpha; // 信息素浓度的影响系数  
private double beta; // 启发式因子的影响系数  
private double pheromoneInitial; // 初始信息素浓度  
private double pheromoneMax; // 信息素浓度上限  
private double pheromoneMin; // 信息素浓度下限  
private double evaporationRate; // 信息素挥发率  
private int antCount; // 蚂蚁数量  
private int maxIterations; // 最大迭代次数  
private boolean maximize; // 是否进行最大化  
private Tour bestTour; // 最优路线  
private double bestFitness; // 最优适应度  
private double[][] distances; // 距离矩阵  
private double[][] pheromones; // 信息素矩阵
```

3、结果展示

多条件约束中，在代码里分别调高固定和行驶成本、时间惩罚成本以及货损成本的权值，可得到不同的配送路径。三者权重分配如表 3-2 所示：

表 3-2 权值分配表

编号	固定和行驶成本权值	时间惩罚成本权值	货损成本权值
1	0.40	0.25	0.15
2	0.15	0.40	0.25
3	0.25	0.15	0.40

配送中心的每辆车的具体运输线路图如图 3-5、3-6、3-7 所示。

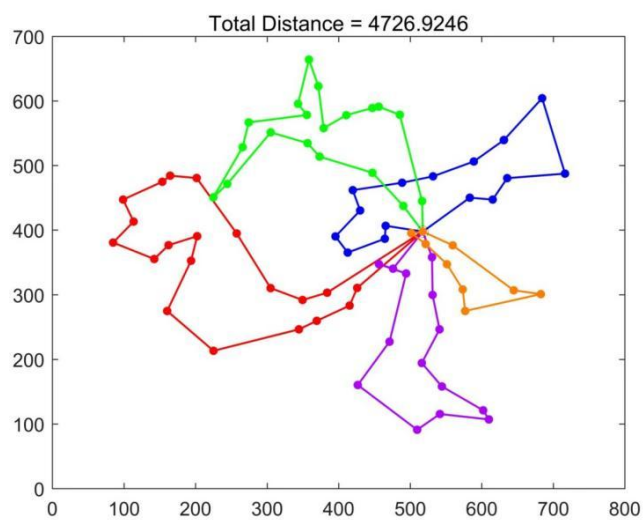


图 3-5 一号路线规划图

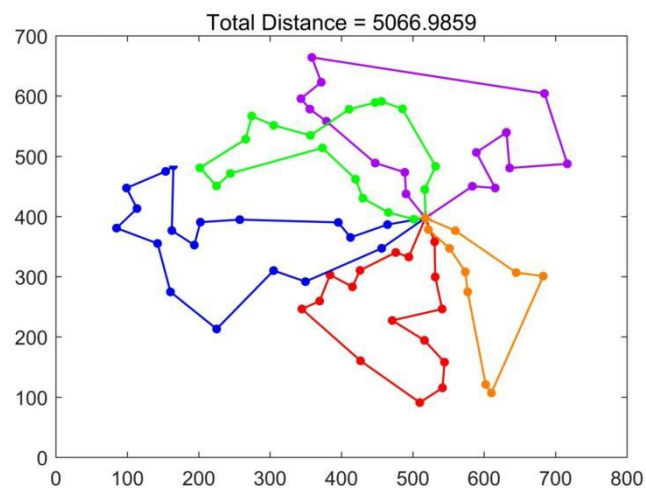


图 3-6 二号路线规划图

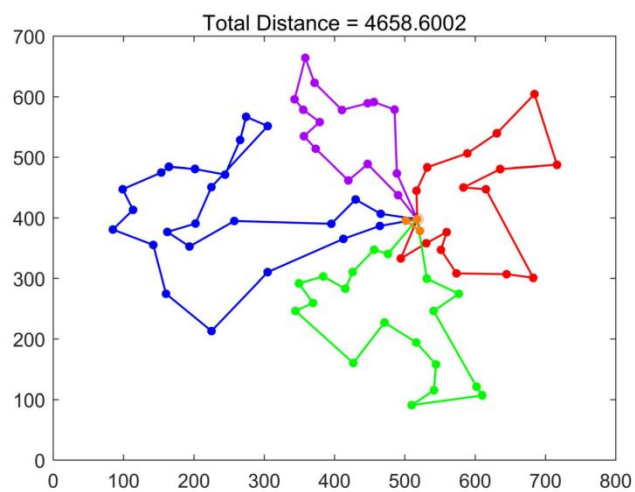


图 3-7 三号路线规划图

3.3.3 信息可追溯

































当前的货物丢失处理大多停留在人工客服层层上报，后续与物流公司多次交互，查询记录，方可得到最后结果。过程时间耗费长，且极易出现货物物流丢失的问题。究其根本就在于物流信息的追溯复杂且涉及隐私，为保障安全性，我们以时间和空间为成本，建立了一套复杂的售后处理机制。而现在的联盟链机制可大幅简化该过程。由于同一联盟下的各组织各节点均有一份相同的账本，使得物流溯源可多服务器同时查询，加快结果的准确性和搜寻的速度。

在本系统中，后端与区块链网络共同组成了整个系统的架构，因此需要考虑如何实现二者之间的数据交互和安全性。这种使用方式便于后端与区块链网络进行安全交互，有效保障了整个系统的安全性和高效性。

进入 WSL，拉取下 fabric 二进制文件与 samples 文件夹，同时利用 docker 指令拉取 fabric 镜像。

名称	修改日期	类型
fabric-samples	2023/4/4 19:00	文件
install-fabric.sh	2023/4/4 18:58	sh_a

而后进入 fabric-samples\test-network 用 ./network.sh up createChannel -ca 拉起网络，注意我们使用了证书授权节点，从而使得后续 SDK 开发方便认证管理。

	Name	Image	Status	Port(s)	Last started	Actions
<input type="checkbox"/>	 dev-peer0.org2.example.com b3d48ceaf498	dev-peer0.org2.example.com	Running		56 minutes ago	  
<input type="checkbox"/>	 dev-peer0.org1.example.com 306cd9cdf10	dev-peer0.org1.example.com	Running		56 minutes ago	  
<input checked="" type="checkbox"/>	 compose	-	Running (7/7)		1 hour ago	  
<input type="checkbox"/>	 cli 5524b8bdf0a1	hyperledger/fabric-tools:late	Running		1 hour ago	  
<input type="checkbox"/>	 peer0.org2.example.com 2fe03f5f2306	hyperledger/fabric-peer:lates	Running	9051:9051 Show all ports (2)	1 hour ago	  
<input type="checkbox"/>	 peer0.org1.example.com a40b9bca26a5	hyperledger/fabric-peer:lates	Running	7051:7051 Show all ports (2)	1 hour ago	  
<input type="checkbox"/>	 orderer.example.com 4db7ad8e2767	hyperledger/fabric-orderer:la	Running	7050:7050 Show all ports (3)	1 hour ago	  
<input type="checkbox"/>	 ca_org1 b40fcd6525b0	hyperledger/fabric-ca:latest	Running	17054:17054 Show all ports (2)	1 hour ago	  

智能合约编写:

选择智能合约编写目录，选择 go 开发，初始化 go 工程:

```
victor@Victor:~/test$ go mod init atcc
touch atcc.go
```

在. go 文件中引入开源库函数依赖包:

```
3 import (
4     "encoding/json"
5     "fmt"
6
7     "github.com/hyperledger/fabric-contract-api-go/contractapi"
8 )
```

函数编写涉及五部分，包括初始化函数和增删改查四个函数，主要在编写的过程中利用了开源 API 的 `ctx.GetStub().PutState(plan.Record, planJSON)` 函数和 `ctx.GetStub().DelState(id)`，分别实现数据上区块和数据信息删除。

```
15 // 定义变量
16 type Plan struct {
17     Record    string `json:"Record"`
18     Driver    string `json:"Driver"`
19     Car       string `json:"Car"`
20     ArrivalTime string `json:"ArrivalTime"`
21     Phone     string `json:"Phone"`
22     Address   string `json:"Address"`
23 }
24
25 // 初始化一个变量组，创建创世区块
26 > func (s *SmartContract) InitLedger(ctx contractapi.TransactionContextInterface) error { ...
27 }
28
29 // 新增一个规划记录到区块上
30 > func (s *SmartContract) CreatePlan(ctx contractapi.TransactionContextInterface, record string, driver string) error { ...
31 }
32
33 // 删除一个规划记录
34 > func (s *SmartContract) DeletePlan(ctx contractapi.TransactionContextInterface, record string) error { ...
35 }
36
37 // 修改规划记录重新上块
38 > func (s *SmartContract) UpdatePlan(ctx contractapi.TransactionContextInterface, record string, driver string) error { ...
39 }
40
41 // 读取出块上的记录
42 > func (s *SmartContract) ReadPlan(ctx contractapi.TransactionContextInterface, record string) (*Plan, error) { ...
43 }
```

接下来我们将智能合约部署到网络中，将自己编写的合约打包并命名为 `plansrecord` 然后部署到网络上。至此联盟链管理员可在开发端进行联盟链网络区块信息的各项管理。

```
+ peer lifecycle chaincode querycommitted --channelID mychannel --name basic
+ res=0
Committed chaincode definition for chaincode 'basic' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vsc, Approvals: [Org1MSP: true, Org2MSP: true]
Query chaincode definition successful on peer0.org1 on channel 'mychannel'
Using organization 1
Querying chaincode definition on peer0.org1 on channel 'mychannel'
Init status: 2, Query committed status on peer0.org2 fails after 1 attempt
+ peer lifecycle chaincode querycommitted --channelID mychannel --name basic
+ res=0
Committed chaincode definition for chaincode 'basic' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vsc, Approvals: [Org1MSP: true, Org2MSP: true]
Query chaincode definition successful on peer0.org2 on channel 'mychannel'
Chaincode initialization is not required
```

API 开发:

在本系统中，后端与区块链网络共同组成了整个系统的架构，因此需要考虑如何实现二者之间的数据交互和安全性。为了保证安全性，我们只允许外部访问两个 API 操作：CreatePlan 和 ReadPlan。其他操作只允许区块链服务器管理人员进行操作，这种方式保证了系统安全性，防止恶意攻击破坏整个网络。

在 API 实现方面，我们选择使用 typescript 编写应用，并引入 express 包，从而实现端口暴露以及请求和响应设置。接收外部请求后，内部编写相应的函数响应。

系统采用 Hyperledger Fabric Gateway 与联盟链网络进行交互。利用 TypeScript 进行智能网关应用的开发，包括建立 gRPC 连接和创立网关连接，具体流程如图 3-8 所示：

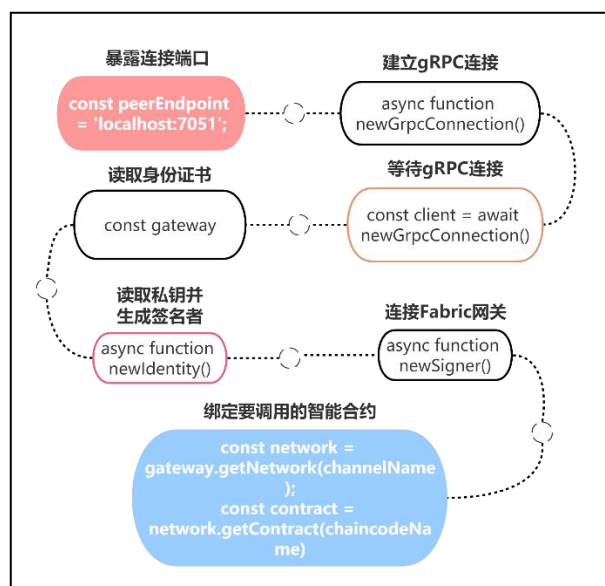


图 3-8 建立 gRPC 连接并与联盟链交互

我们将联盟链的 7051 端口连接后，启用主机的 8091 端口进行外部监听。运行代码：

```

victor@Victor:~/blockchain/fabric-samples/asset-transfer-basic/gateway$ yarn run start
yarn run v1.22.19
$ node dist/app.js
listening at http://localhost:8091
  
```

对于接收到的请求，我们设置了两种请求模式，分别对应 ReadPlan 和 CreatePlan 操作。这种使用方式便于后端与区块链网络进行安全交互，有效保障了整个系统的安全性和高效性。

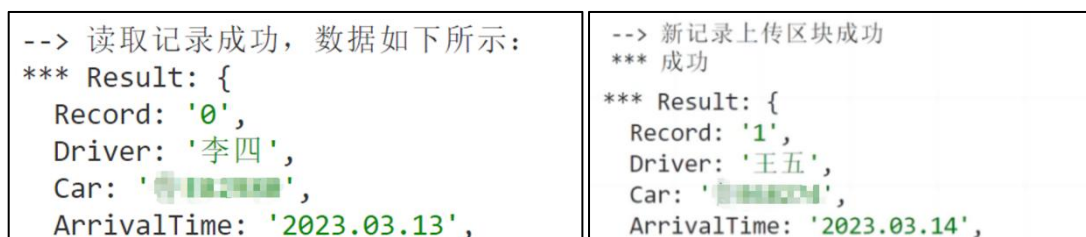


图 3-9 ReadPlan 和 CreatePlan 运行结果图

3.4 应用层实现

3.4.1 网页 Web 实现

网页端主要基于 Vue2.0 和 SpringBoot 前后端分离搭建而成，使用 php 实现网页与云数据库的连接与通信，前端利用 ajax、html 等实现交互动态网页。

后端使用 Java 语言编写，采用 SpringBoot 搭建的 ssm 轻量级框架，使用 Maven 作为项目管理工具，通过 pom.xml 中的配置，可以从仓库获取到想要的 jar 包，并且可以表示项目间的继承依赖等关系。前后端通过 JSON 数据格式进行通信，约定好接口，达到了前后端分离的标准。

数据库使用关系型数据库并采用主从设计，数据库的隔离级别选择的是可重复读，通过一致性视图保障事务，考虑到读多写少的业务情况，也加入了缓存的设计，保障了可用性。运行环境可以支持任意，可迁移性强，可以在独立服务器或者虚拟机上部署，处处运行。并且在全系统中赋予了日志记录和权限控制的模块，使得系统更加完整。及时定位报错点，更易于维护。权限的分级与控制约束了误操作的情况发生，并且十分严谨。

最终在利用前后端实现了账号关系、权限分发、司机车辆信息的增删改查、配送申请、车辆监控查看实时数据、数据可视化、本系统货车分布图、多目的地路径规划、配送信息写入联盟链、反馈处理等功能。

3.4.2 小程序实现

小程序基于 Vant3 架构搭建而成，面向司机与监管人员。使用 PHP 和 wx.request 完成与服务器的连接与通信，为司机提供天气预报、实时车辆数据监控、车辆分布图、路径规划导航、疲劳驾驶检测、货运信息写入联盟链、反馈建议等功能。

第 4 章 应用效果

4.1 数据采集上传实现效果

TFT 屏幕显示与云平台数据显示：

首先是网关终端的数据实时上传云平台 and 屏幕显示功能。启动设备查看传感器测量的各个数据，在屏幕上实时显示，当网关节点连接到车载 WiFi 时，会自动将信息上传至云平台 and 云服务器，数据自动同步。如图 4-1 所示，可以看到终端屏幕实时显示的数据与云平台中接收到的数据一致，并且实时更新，即网关节点上传数据的功能正常。



图 4-1 系统实物数据展示

数据名称	最新数据	最后更新时间
TM 车厢温度	23	2023-04-18 10:10:59
HMI 车厢湿度	47	2023-04-18 10:10:59
MQ 烟雾浓度	4	2023-04-18 10:10:59
JD 车辆经度	114.281111	2023-04-18 10:10:59
P 驾驶位检测	0	2023-04-18 10:10:59
WD 车辆纬度	30.274444	2023-04-18 10:10:59
F 火焰检测	1	2023-04-18 10:10:59
R 人车货是否检验	1	2023-04-18 10:10:59
T 司机体温	36	2023-04-18 10:10:59

图 4-2 云平台数据展示

车厢温湿度检测：

通过改变车厢内温湿度，测试感知层和应用层数据是否实时同步变化。经过测试可知，温度和湿度统一为 27°C，99%，如图 4-3 所示，该项功能正常。



图 4-3 测试温湿度

车内人员检测：

通过检测车厢内有无人员以及乘车人温度，检测感知层和应用层数据是否实时同步变化。经过测试可知，检测出有司机且体温正常，如图 4-4 所示，该项功能正常。



图 4-4 测试驾驶位检测和驾驶员温度

火焰检测:

利用火焰报警器观察前端应用是否会报火警,检测感知层和应用层数据是否实时同步变化。经过测试可知,当检测到火星后会暴红色警报,提醒及时处理,如图 4-5 所示,该项功能正常。



图 4-5 测试火焰检测

RFID 识别与 Lora 通信检测:

RFID 终端的读卡能力及实时显示功能,我们对已经录入系统的驾驶员、车辆、货物进行刷卡识别,如图 4-6 所示,可以看到屏幕显示的卡号与实际卡号一致。再使用管理员卡和未录入系统的卡进行刷卡,屏幕分别显示管理员权限和错误提示,可见 RFID 识别功能部分正常,能够正确识别不同的卡号。

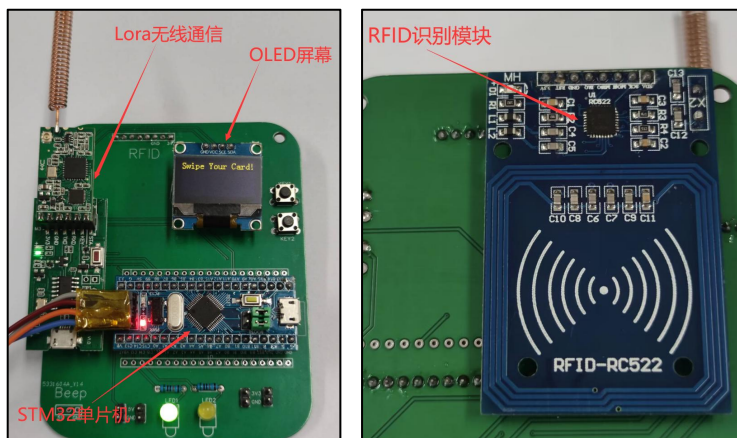


图 4-6 RFID 终端实物图

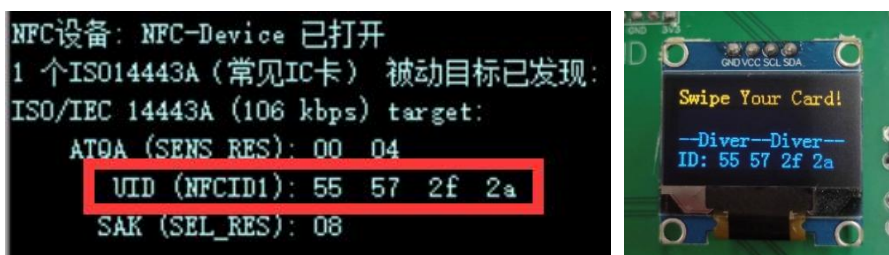


图 4-7 驾驶员卡号信息及刷卡图 1

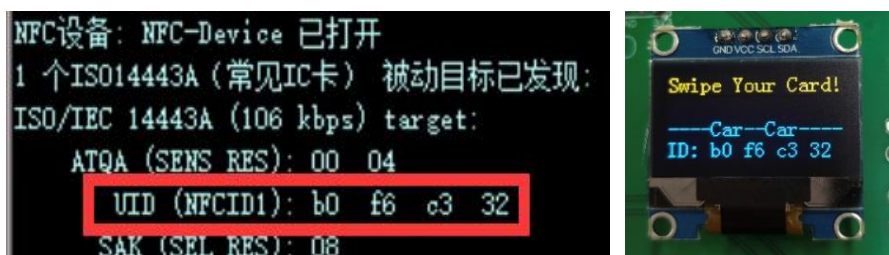


图 4-8 车辆卡号信息及刷卡图 2



图 4-9 货物卡号信息及刷卡图 3

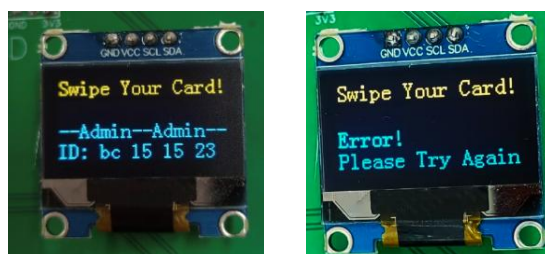


图 4-10 管理员及其他卡片测试图

4.2 疲劳驾驶识别实现效果

监测驾驶员驾驶时行为状态，如闭眼、打哈欠等疲劳驾驶行为。测试摄像头能否准确录入视频，树莓派能否准确识别驾驶员的驾驶状态。

如图 4-11 所示，当测试者困倦长时间闭眼时，系统准确识别出困倦状态，并提示“Drowsiness Alert”。当测试者打哈欠时，系统准确识别出打哈欠状态，并提示“Yawn Alert”。由此可见，驾驶行为识别部分功能正常。



图 4-11 疲劳驾驶-闭眼测试图



图 4-12 疲劳驾驶-打哈欠测试图

4.3 多目的地路径规划实现效果

首先根据多个客户的需求填写待配送信息，如图 4-13 所示。

需求13	收货人: 张六	花卉植物	收货地址: [模糊地址]
	2023-03-01 15:27 ~ 2023-03-01 17:27		联系方式: [模糊电话]
删除此需求			
需求14	收货人: 刘三	冷藏食品	收货地址: [模糊地址]
	2023-03-01 12:32 ~ 2023-03-01 13:32		联系方式: [模糊电话]
删除此需求			
需求15	收货人: 王五	冷藏食品	收货地址: [模糊地址]
	2023-03-01 14:32 ~ 2023-03-01 16:32		联系方式: [模糊电话]
删除此需求			
需求16	收货人: 卜四	化学药品	收货地址: [模糊地址]
	2023-03-01 08:32 ~ 2023-03-01 10:32		联系方式: [模糊电话]
删除此需求			
<div>Cancel OK</div>			

图 4-13 填写多目的地配送客户需求

系统会根据多条件约束路径优化算法自动生成路线安排，并分配给不同司机不同货车进行运输，如图 4-14 所示。

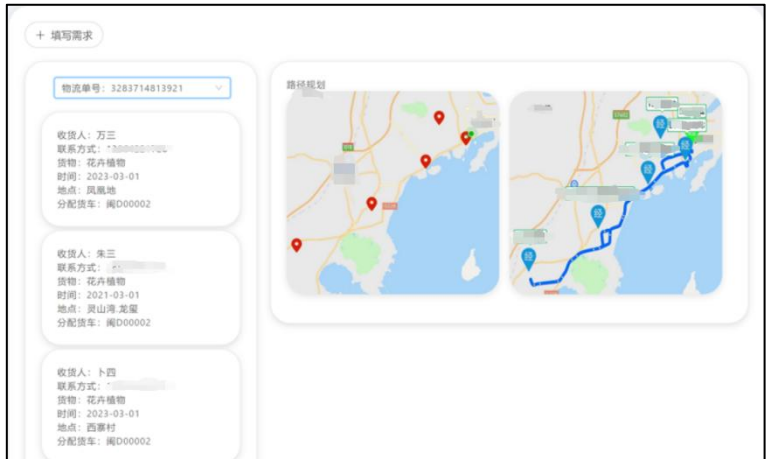


图 4-14 系统根据算法自动生成最优路径

同时可检测到配送列表自动生成配送记录与状态显示，如图 4-15 所示。

司机	车牌号	客户电话	客户地址	注意事项	预计送达	当前状态	操作
卜四	闽D00002	13905911111	多个		2023-03-01T03:38:55.092Z	正在运输	编辑 配送
赵三	闽D00003	13905911111	多个	防止高温	2023-03-01T03:40:49.380Z	正在运输	编辑 配送
李四	鲁A00001	13905911111	多个		2023-03-01T03:39:52.161Z	正在运输	编辑 配送

图 4-15 配送信息详情

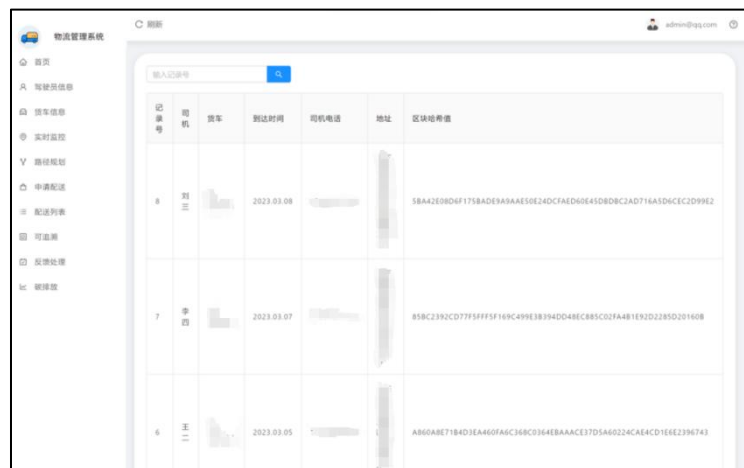
4.4 快递信息可追溯实现效果

快递信息可追溯功能：

- ① 检索已经记录的数据，是否在用户界面中展示。
- ② 检索未记录的数据，系统是否自动与联盟链网络进行交互，并将记录数据呈现出来。

实现效果：

- ① 打开前端，进入可追溯界面，观看历史查询记录。如图 4-16 所示，记录包括记录号、司机、货车、到达时间、司机电话、地址以及联盟链块哈希值。

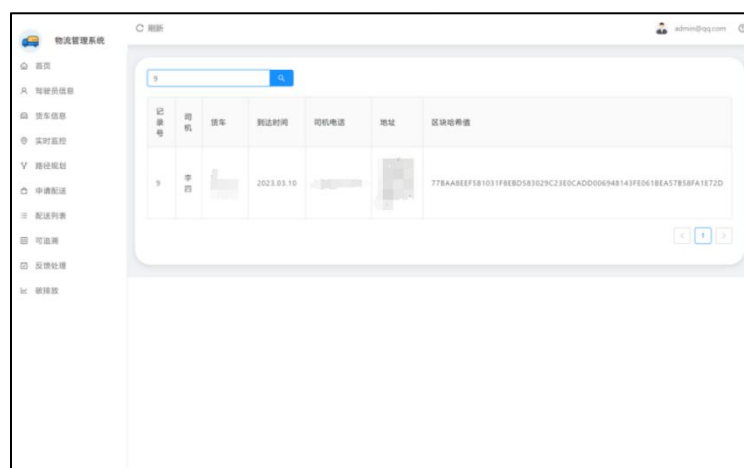


The screenshot shows a web application titled '物流管理系统' (Logistics Management System). On the left is a sidebar menu with options like '首页' (Home), '联盟链信息', '货车信息', '实时监控', '路径规划', '申请配送', '配送列表', '可追溯', '反馈处理', and '被追踪'. The main area features a search bar '输入记录号' and a table of records. The table has columns for '记录号' (Record ID), '司机' (Driver), '货车' (Truck), '到达时间' (Arrival Time), '司机电话' (Driver Phone), '地址' (Address), and '区块链哈希' (Blockchain Hash). Three records are visible, each with a corresponding hash value.

记录号	司机	货车	到达时间	司机电话	地址	区块链哈希
8	刘三		2023.03.08			5BA2E9D4F175BAD5A9AE5E24DCE4D6E45D8D6C3AD716A5D6CEC2D99E2
7	李四		2023.03.07			858C2392CD77F5FF5F169C49E38394DD48EC85C02FA81E92D285D291608
6	王二		2023.03.05			AB60A8E71B4D3EA460FA6368C9364E8AAACE37D5A60224CAE4CD1E6E2396743

图 4-16 信息可追溯历史记录

② 检索未曾搜索过的记录，如图 4-17 所示，会快速与联盟链网络交互并显示出记录。

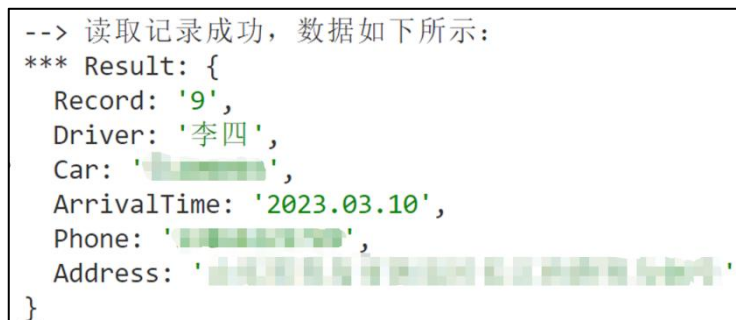


The screenshot shows the same web application as Figure 4-16, but the search bar now contains the number '9'. The table displays a single record for record ID 9, with the driver '李四' and arrival time '2023.03.10'. The '区块链哈希' (Blockchain Hash) is '77BA8EEF581031F8EBD58329C23E0CAD006948143FE0618EA57858FA1E72D'. Navigation buttons for the table are visible at the bottom right.

记录号	司机	货车	到达时间	司机电话	地址	区块链哈希
9	李四		2023.03.10			77BA8EEF581031F8EBD58329C23E0CAD006948143FE0618EA57858FA1E72D

图 4-17 可追溯信息检索

③ VSCode 检测到前端与联盟链交互，并在命令行显示出交互内容，如图 4-18 所示。



The screenshot shows a terminal window with the following text:

```
--> 读取记录成功，数据如下所示：
*** Result: {
  Record: '9',
  Driver: '李四',
  Car: '[REDACTED]',
  ArrivalTime: '2023.03.10',
  Phone: '[REDACTED]',
  Address: '[REDACTED]',
}
```

图 4-18 联盟链网络内信息详情

4.5 前端 Web 实现效果

1、登陆界面



图 4-19 登陆界面

登录界面实现了管理员登录的功能。管理员登录有两种登录方式：账号密码登录和验证码登录。

2、首页



图 4-20 首页

首页由多个模块组成。顶部由刷新按钮和登录用户设置构成，其中登录用户设置可以进行退出用户登录的操作。主模块包含管理员信息显示模块、在线司机数模块、在线货车数模块、事项速办模块、运输物品分类模块、季度运输物品数量变化模块、货车分布模块。

管理员模块显示管理员头像，管理员 ID 以及管理员的权限。在线司机数模块、在线货车数模块实时动态更新显示在线司机数和在线货车数。事项速办模块显示最新的事项通知，管理员可以根据事项通知反馈做出相应的处理。将处理结果告知司机。

运输物品分类模块，这里显示运输物品的分类以及所占的比重。季度运输物品数量变化模块显示运输物品数量在四个季度的不同的数量，随季度的变化情况。货车分布模块显示现在地图上所有的货车的分布情况，在地图上所处的位置。

3、驾驶员信息界面

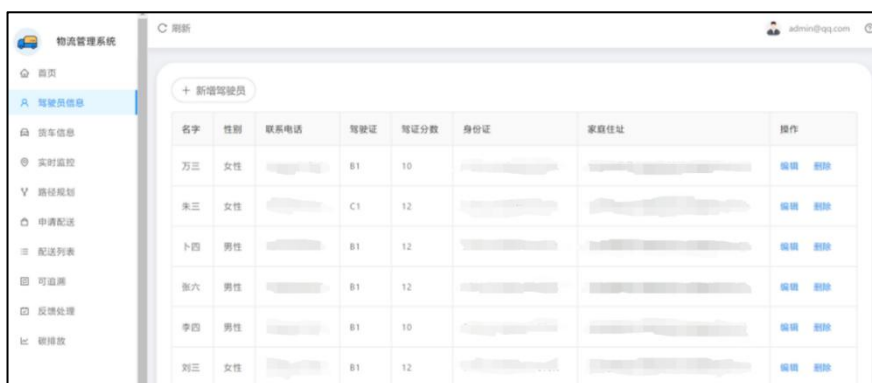


图 4-21 驾驶员信息界面

这部分显示的展现的是驾驶员的信息情况，包括驾驶员的名字，性别，联系电话，驾驶证，身份证，家庭住址。管理员可以对驾驶员的信息进行修改和删除，此外还可以新增驾驶员。

4、货车信息界面

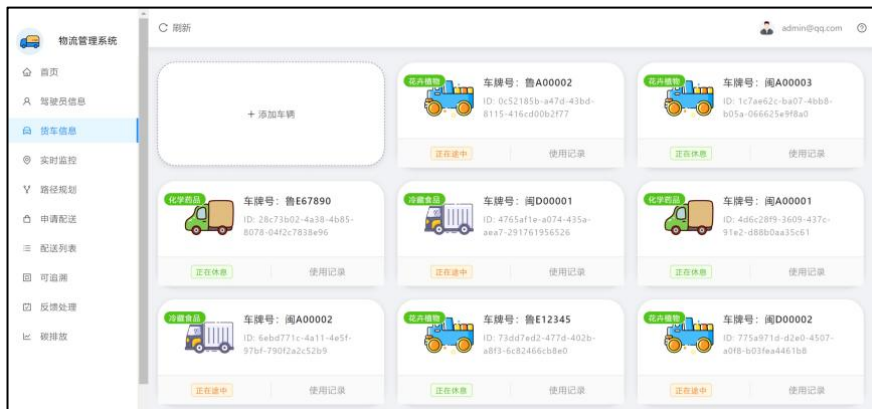


图 4-22 登陆界面

这部分显示的展现的是货车的信息情况。包括货车的类型包括冷藏食品、化学药品、花卉植物，货车的车牌号，货车的 ID，以及货车的状态即货车是在正

在途中还是正在休息，同样也可以查看货车的使用记录，同时管理员可以进行车辆的添加。

5、实时监测界面



图 4-23 实时监测界面

这部分显示的展现的是实时监测的数据，包括车内的温度，车内的湿度，烟雾浓度，以及对火焰的检测。这里利用 OneNET 以及 API 技术，通过温湿度传感器、烟雾传感器、火焰传感器将数据存储至云数据库，前端调用后端，后端调用数据库，将数据展示到前端页面。这部分还展现了 12 个小时内每个小时的监测的温度和湿度数据，实时展现了司机的状态，显示车内是否有人员存在，以及司机现在的体温。

6、申请配送界面

这部分展现的是申请配送的界面，在这里可以填写申请配送相关的信息，确认配送的信息，完成申请的信息。

Figure 4-24 shows the application for delivery interface. The sidebar on the left contains navigation links: 首页, 驾驶员信息, 货车信息, 实时监控, 路径规划, 申请配送 (highlighted), 配送列表, 可追溯, 反馈处理, and 报错. The main content area is titled '申请配送'. It features a three-step process: 1. 填写申请信息 (Fill in application information), 2. 确认配送信息 (Confirm delivery information), and 3. 完成申请 (Complete application). The form includes fields for selecting a driver (司机) and vehicle (车辆), and a date picker for the estimated delivery time (预计交货时间). There is a checkbox for '加急处理' (Urgent processing). Below, there is a section for '注意事项' (Notes) with checkboxes for '冰柜冷藏' (Ice chest refrigeration), '注意易碎' (Note fragile), and '防止高温' (Prevent high temperature). The form ends with fields for '客户电话' (Customer phone) and '客户地址' (Customer address), followed by a '下一步' (Next step) button.

图 4-24 申请配送界面

管理员首先填写申请信息，包括对司机的选择，对运输车辆的选择、预计交货时间、是否进行加急处理、注意事项、客户电话、客户地址。填写完申请信息之后，管理员需要对配送信息进行确认。



图 4-25 确认配送界面

接下来管理员需要对配送信息进行确认，如果配送信息正确，就提交进行下一步，如果配送信息有误，就返回上一步进行修改。

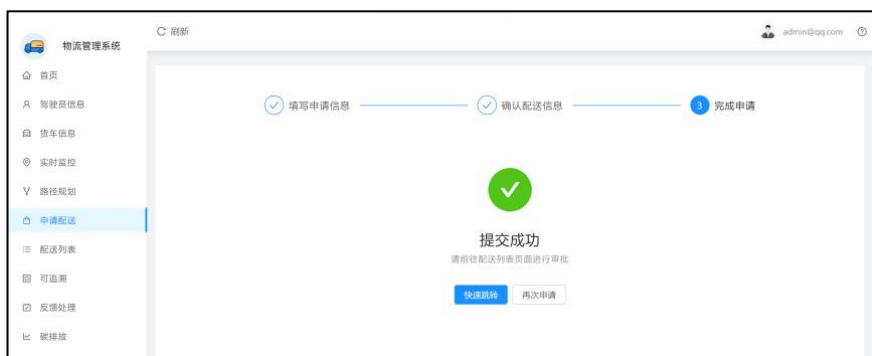


图 4-26 完成申请界面

最后完成申请，显示提交成功。

7、路径优化界面

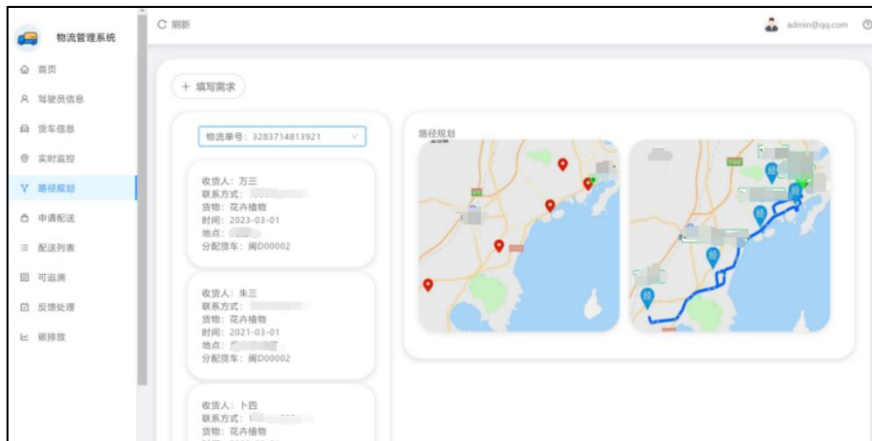


图 4-27 路径优化界面

管理员根据多用户的需求填写需求表单，然后系统根据路径优化算法，将多个需求划分成不同组合分配给不同货车并分配对应的订单号。每个物流单号下显示着该组的客户需求，同时根据配送先后顺序由上到下排列。右侧为配送的具体路线。同时该物流单号下的信息可下发给出货人员，出货人员按照单号倒序将货物由内到外进行排放，降低卸货时的困难。

8、配送列表界面



图 4-28 配送列表界面

这部分显示的展现的是申请配送的界面，显示配送信息和状态，包括司机姓名，货车的车牌号，客户电话和客户地址，以及在配送过程中的注意事项，预计送达的时间，并且可以查看这辆货车的配送状态，例如正在配送、配送完成的状态。管理员可以在此查看到配送的相关信息，并对其进行操作包括修改客户电话以及客户地址、删除配送信息。

9、可追溯界面

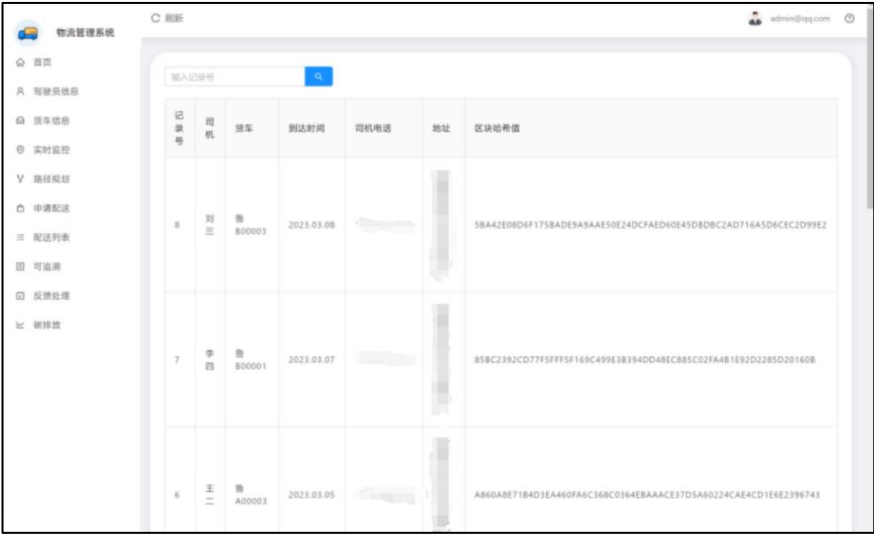


图 4-29 可追溯界面

这部分展现的是可追溯界面，这部分展示了货车配送过程的信息，包括记录号、司机姓名、货车编号、到达时间、联系号码、地址信息。管理员可以在此查看到每次的配送过程，在发生事故之后可以及时追溯到相关配送信息。

10、反馈处理界面



图 4-30 反馈处理界面

这部分展现的是反馈处理界面，展示了司机在配送过程中反馈的信息。包括具体的时间、反馈用户的用户名、反馈的内容以及处理的状态。管理员可以在此查看到司机反馈的信息，及时处理司机在行驶过程中遇到的问题，保证高效安全的进行物流的配送。

11、碳排放界面

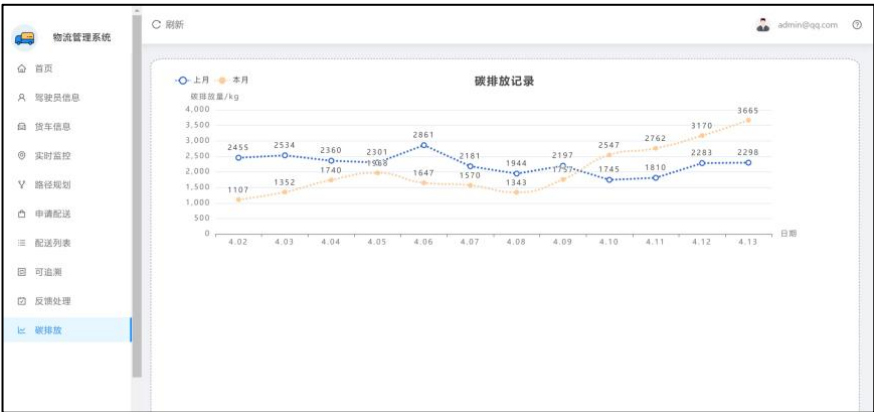


图 4-31 碳排放界面

这部分展现的是碳排放界面，为路径优化前后碳排放量的变化。路径优化使行驶距离更短，车辆行驶更加高效，从而减少了油耗和碳排放量。管理员可以根据碳排放量的变化量来查看路径优化对碳排放变化的影响。

4.6 小程序实现效果

1、用户登录界面、首页



图 4-32 登陆界面



图 4-33 首页

扫码进入小程序，首先是登陆界面，输入账号和密码即可登录。进入首页后，顶部为系统提示和消息栏，下半部分为天气预报、车辆实时数据查询、车辆分布图、路径规划导航、运输信息上传联盟链、驾驶员疲劳驾驶检测按钮。点击相应的按钮即可跳转到与之对应的页面。

2、天气预报查询、车辆实时数据查询



图 4-34 天气查询



图 4-35 车辆实时数据

为了帮助快递运输驾驶员了解当地的天气情况,做好防范措施,如减速行驶、调整车速、更换轮胎等等,我们在这里设计了天气预报查询的功能,帮助其提前做好应对措施,以保证货物安全,避免发生交通事故。

车辆实数据监控页面提供快递运输车当前的实时经纬度、司机体温,车厢内的温度、湿度、烟雾浓度、是否有火焰等数据,所有数据均为硬件采集端实时获取并通过 WiFi 网关端上传到云平台 and 云服务器,小程序实时显示,所有数据实时更新。

3、路径导航优化、货车分布地图

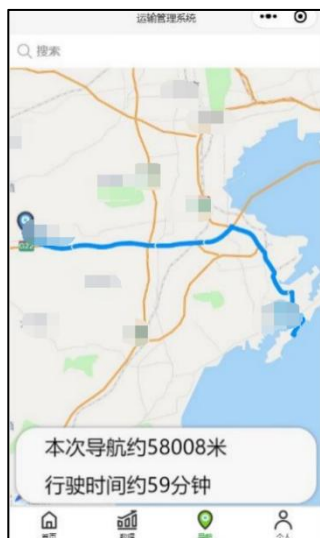


图 4-36 路径导航

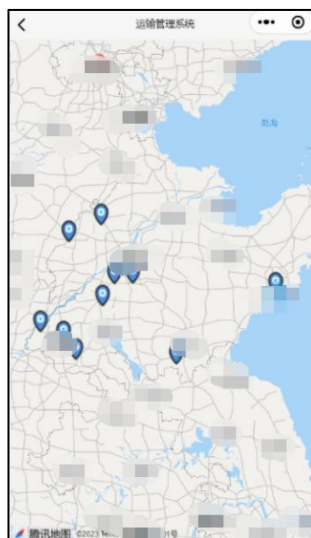


图 4-37 车辆分布

在路线规划页面,这里的实时位置获取使用了微信提供的 `wx.getLocation` 接口,使用此接口获取实时位置的经纬度,再通过腾讯位置服务提供的逆地址解析接口解析出经纬度对应的地址信息,用户也可以查看规划好的路线。

点击首页中的车辆分布按钮可以跳转到此页面,查看使用该系统的同城车辆,方便查找附近车辆寻求帮助。

4、信息上链、疲劳驾驶检测



图 4-38 信息上链

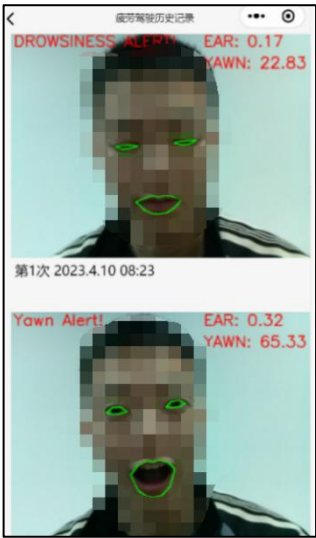


图 4-39 疲劳检测

信息上链功能主要是填入配送时间、司机账号、手机号、使用的车辆信息、出发地、目的地、货物类型等信息，点击上传信息至联盟链即可将该信息打包上传至 Fabric 联盟链网络，方便以后查看和追溯。疲劳检测页面可以查看来自树莓派拍摄到的驾驶员疲劳状态的照片和对应的时间点，方便查看历史驾驶记录。

5、反馈建议、个人界面



图 4-40 反馈建议



图 4-41 个人页面

此页面用户可以向系统管理员反应自己遇到的问题，管理员会在后台看到反馈，及时更改相关问题。当用户有疑问时也可以通过此页面和平台工作人员进行联系，我们将协助用户解决相关问题。

第5章 创新与特色

本作品在快递运输的应用场景下，提出了智慧快递运输系统的设计方案，实现了实时监控、在途跟踪、多点路径规划、信息追溯、用户隐私保护等功能。作品的创新与特色点如下：

5.1 改进的遗传-蚁群融合算法

本作品针对快递运输车辆需要将快递配送到一个地区多个快递点情况，采用改进的遗传-蚁群融合算法实现多目的地路径规划。

改进的遗传-蚁群融合算法综合利用了遗传算法的全局随机搜索能力和蚁群算法的优化特性，具有强大的搜索能力，可以在较短时间内找到最优解。该算法还能够在路径规划过程中能够快速收敛，避免陷入局部最优解。此外，该算法综合考虑了货车的固定和行驶成本、时间惩罚成本以及货损成本等多种因素，在保证运输距离最短、时间最短的情况下，尽可能降低实际的运输成本和时间，提高了快递配送的效率和质量，具备较高的实际应用价值。

5.2 基于层次身份的加密和环签名

本作品针对目前快递用户信息隐私保护方案中没有考虑快递站点数量庞大的情况，也未确保快递用户信息密文的有效性的情况，采用基于层次身份的加密和环签名解决上述问题。

基于层次身份的加密算法能够通过多层域 CA 来完成大量快递站点的密钥生成。环签名不仅能够证明消息的可靠性，还能够保护签名者身份。区块链技术能够确保交易数据不被篡改。因此，我们设计了一个基于层次身份的加密算法，并利用该算法对快递用户信息进行分段加密。基于层次身份的密钥生成方式，我们设计了一个环签名算法，由揽收站点使用该算法对快递用户信息密文进行环签名，然后将密文和环签名上传到区块链中，由区块链共识节点验证环签名是否有效。这种算法有效的解决了用户隐私泄露的问题，保障了用户个人信息的安全性。

5.3 快递运输过程实时监控

本作品针对目前快递中无法及时跟踪邮政小包的转运进度,在对特殊物品运输时环境容易造成物品损坏的情况,采用实时监控解决上述问题。

本系统利用多传感器融合技术,使用温度、湿度、烟雾等传感器实时采集车内环境的数据,利用 GPS 定位系统实时定位车辆位置。传感器将货车各个参量数据实时上传至云端,并通过应用层进行展示和管理,使管理员可以及时监测车内的环境和状态,并及时发现车内异常情况。实现物流流程高度信息化,提高物流运营效率和管理水平。

第 6 章 未来展望

6.1 作品成果

1、项目成果

项目从当前快递运输实际情况出发，开发基于 AIoT 和区块链的智慧快递运输管理系统，Adaboost 算法、多传感器融合技术实现了**实时监控**和**司机疲劳检测**，保障了运输过程的安全。项目采用了改进的遗传-蚁群融合算法，实现了**多目的地路径规划**，提高了运输以及管理效率。此外，项目还采用了基于层次身份的加密和环签名，**保障了用户的个人信息安全，防止用户隐私泄露**。

2、科技成果

项目进行期间，团队正在**申请软件著作权与专利**，利于形成专业技术壁垒，具有推广的技术优势与支撑。

6.2 应用前景

本项目提出的智慧快递运输管理系统从快递运输的整个流程出发，通过 AIoT、区块链等技术，解决了运输过程车内状态无法实时监测、快递运输配送效率低、用户隐私安全难以保障、快递信息难追溯等问题，帮助快递企业能够更好的满足管理快递系统的需求。此外，本项目提出的路径优化算法，提高了配送效率，减少了企业的运输成本，具有广阔的应用前景。

随着我国经济的快速发展，快递行业的业务量逐年增加并且快递已经在人群中普遍存在，而配送作为快递企业最关键的环节，在企业的配送效率、服务质量、竞争力优势、市场占有规模和客户满意度方面有着非常大的影响。因此，智慧快递运输管理系统可以有效提高企业对快递运输的管理，对于提高企业管理效率，快递配送效率发挥着十分重要的作用。

国家相关部门相继出台了大量政策文件促进为了快递服务业的快速发展，2021 年 3 月国家发改委发布的《中华人民共和国国民经济和社会发展第十四个五年规划和 2035 年远景目标纲要》首次高频次提及要将物流与供应链发展提升新高度；2021 年 12 月国家邮政局发布的《“十四五”快递业发展规划》文件，也指出要将快递服务水平提升到新的高度；实现到 2025 年快递服务总体满意度提高至少 3.3 分，全国重点区域快递服务 72 小时准时率提升 2.9%。本项目提出

的快递物流管理系统，帮助快递行业提高了快递运输效率，同时也提高了快递的服务水平，符合国家快递行业的发展政策，具有很好的现实应用前景。此外，本项目通过采用改进的遗传-蚁群融合算法，进行多点路径规划，有效的减少了燃油量，降低了碳排放，符合国家“双碳”的政策要求，是国家能源安全稳定与可持续发展的选择。

因此，就**现阶段**来看，本项目作为线上平台，具有很高的实用性，就**长远**来看，项目未来可扩展应用于其他运输行业中，具有一定商业潜力。