仓储搬运设备远程故障诊断系统
详细设计说明书

东南大学自动化学院

2016 年 8 月 31 日

版本历史

版 本 号	更 改 描 述	更 改 日 期	修 订 人

目 录

1，引言

1.1 编写目的

本文档为“基于物联网的仓储搬运设备远程故障诊断系统”中上位机监控软件的详细设计文档。经过认真的需求分析与系统论证之后确定了最终设计方案，包含软件界面设计，各个功能模块的实现、通信流程、数据处理过程、数据存储过程等。

1.2 项目背景

随着工业技术的迅速发展，越来越多的自动化设备应用在了我们的日常生产生活当中，不断加快了工业化的进程。工业搬运车辆也被广泛应用在了港口、车站、机场、工厂车间、仓库、流通中心和配送中心等多个场所，叉车作为物料搬运设备中的主力军，在企业的物流系统中扮演着非常重要的角色。叉车主要发展于第二次世界大战期间，并于上世纪 50 年代逐渐开始在中国生产制造，而随着中国经济的快速发展，大部分企业的物料搬运工作已经脱离了原始的人工搬运模式，取而代之的是以叉车为主的机械化搬运。

仓储叉车是一种用于在仓库内搬运货物的叉车，其应用不仅能够节省人力物力，减少搬运的成本，还能进一步提高操作的安全性，避免事故的发生。大部分的仓储叉车都是以电动机驱动的，因其车体紧凑、移动灵活、自重轻和环保性能好等显著特点在仓储业中得到了广泛应用。

叉车等仓储搬运设备在使用过程中，不可避免的会出现各种类型的故障。现阶段生产过程中叉车出现故障后，一般还是采用传统的维修方法，即故障较为简单易诊断时，由用户自行在厂家指导下进行一些基础的维修工作；故障较为复杂难以诊断时，由厂家派出专业的维修人员前往现场对设备进行故障排除检查以及维修工作。厂家现场维护不仅会造成人力和物力上的浪费，增加维护成本，还会因为维修上的及时性影响用户的正常作业，导致生产无法正常运行。另一方面，当用户自行维护时，由于其对设备本身不够熟悉，也可能出现维护不当等问题，造成搬运设备的寿命下降。

物联网、云计算等技术的快速发展，为仓储搬运设备智能维护提供了新的解决方案。通过智能终端采集搬运设备操作和运行状态信息，并汇集到企业数据中心，通过故障分析与决策，可以实现搬运设备的远程监控，指导用户进行设备维护，从而可以提高搬运设备维护的智能化程度。因此，仓储搬运设备的远程监控的实现具有十分重要的意义。

1.3 术语定义及说明

文档中采用的专门术语的定义及缩略词简要如下：

WIFI：Wireless Fidelity， 基于 IEEE 802.11b 标准的无线局域网

GPRS:General Packet Radio Service，通用分组无线服务技术

1.4 参考资料

[1] 黄迪. 物联网的应用和发展研究[D].北京邮电大学,2011.

[2] 刘若冰. 物联网的研究进展与未来展望[J]. 物联网技术, 2011, 05:58-62.

[3] 周璞, 胡杰强, 龚军. 基于 GPRS 的电动汽车远程监控系统的开发设计: 第九届河南省汽车工程技术研讨会, 中国河南许昌, 2012[C].

[4] 朱炫鹏. 基于 GPRS 的远程监控系统的应用研究[D]. 东南大学,2005.

[5]刘佳. 基于 B/S 结构的远程故障诊断系统研究与实现[D]. 南华大学, 2012.

[6] 尹洪珠. 基于 Internet 的机械设备的远程故障诊断中若干问题的研究[D]. 大连理工大学, 2002.

[7]朱顺华. 基于 GPRS 的汽车远程故障诊断系统中车载终端的研究设计[D]. 国防科学技术大学, 2006.

2，设计概述

2.1 任务和目标

本监控软件的目标是实现对仓储搬运设备进行全方位的远程监控与故障诊断，包括：设备实时状态监控、设备故障实时报警及分析、历史故障信息统计与

分析、历史状态信息及故障信息查询等。

监控软件分为三部分，即服务器软件、局域中心软件和单机诊断软件。服务器软件主要完成对从智能终端发送过来的状态信息、故障报警信息进行接收和存储处理，并实现对叉车信息的实时更新及故障报警，同时服务器软件还提供历史信息查询、故障统计、维修信息查询及添加等功能；局域中心软件可实现对多台智能终端联网功能，统一发送叉车的状态及故障信息数据，提供人机交互界面；单机诊断软件主要用于设备复杂故障处理，需要专业维修人员现场连接故障诊断智能终端，方便维修人员在现场对叉车状态信息进行实时查询与分析，便于快速定位故障。

2.2 操作系统配置

操作系统框架为 Microsoft .NET Framework 4.5，硬件和软件配置如下图 1 所示：

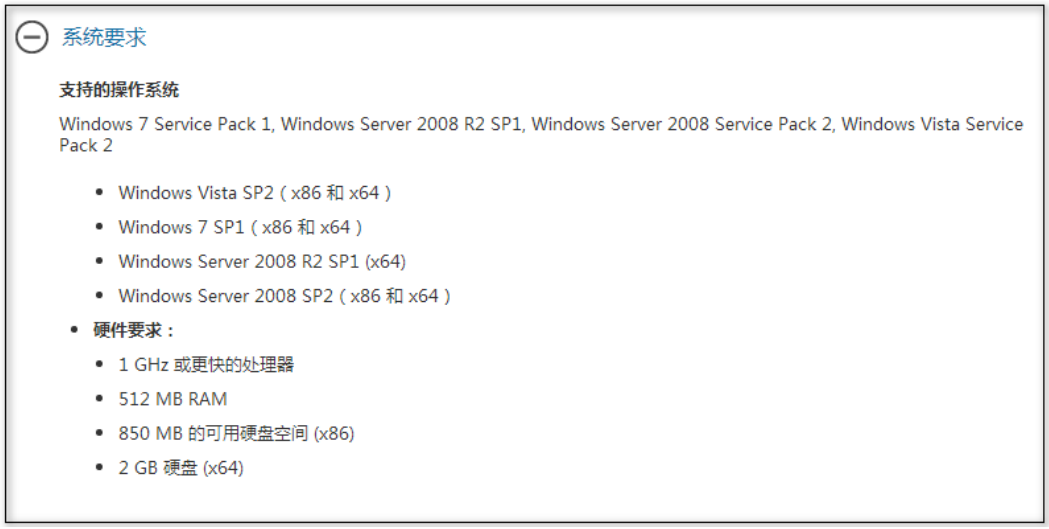

图1 操作系统硬件和软件配置

2.3 条件与限制

智能终端的 WIFI 或 GPRS 模块在与服务器通信时，需要服务器有固定 ip，因此在搭建服务器时要申请外网固定 IP。

2.4 开发工具

综合考虑软件的开发效率及稳定性要求，拟采用 windows 下的 visual stdio 2015 .net framework 框架进行开发，开发语言为 C#,数据库采用 MySQL 进行数据存储与查询。

3，软件需求分析

3.1 服务器软件需求分析

- 服务器功能需求分析

服务器软件是监控软件最重要的组成部分，也是所有数据的汇总中心，功能较为复杂，经分析，服务器功能需求主要包括以下几方面：

- ✧ 数据接收处理与存储
- ✧ 在线叉车状态显示
- ✧ 故障报警显示
- ✧ 历史故障信息显示
- ✧ 历史状态信息查询
- ✧ 故障信息统计
- ✧ 故障统计

- 服务器性能需求分析

服务器软件具有良好的性能，是整个系统正常运行的前提。经分析，服务器性能需求包括以下几方面：

- ✧ 能够长时间稳定运行，不能因为异常问题发生宕机。
- ✧ 接收至少一万台智能终端的连接，并能正常运行。
- ✧ 不发生数据拥堵，内存泄漏等问题。

3.2 单机诊断软件需求分析

- 单机诊断软件功能需求分析

单机诊断软件是监控软件重要的组成部分，方便用户在现场进行故障诊断和维修。因此，根据使用场景分析，单机诊断软件功能需求主要包括以下几方面：

- ✧ 数据接收处理
- ✧ 叉车状态显示
- ✧ 故障报警显示

● 单机诊断软件性能需求分析

单机诊断软件需要帮助用户完成现场故障诊断的功能。因此，单机诊断软件性能需求包括以下几方面：

- ✧ 通过串口连接智能终端后能够准确获取叉车当前所有信息。
- ✧ 不发生数据拥堵，内存泄漏等问题。

3.3 局域中心软件需求分析

● 局域中心软件功能需求分析

局域中心软件接收对应的智能终端的数据信息，可以看作是小型的服务器软件，功能较为简单。经分析，局域中心软件功能需求主要包括以下几方面：

- ✧ 数据接收处理
- ✧ 叉车状态显示
- ✧ 故障报警显示

● 局域中心软件性能需求分析

局域中心软件方便搬运设备客户使用，因此仅对稳定性有较高的要求。经分析，性能需求包括以下几方面：

- ✧ 能够长时间稳定运行，不能因为异常问题发生宕机。
- ✧ 不发生数据拥堵，内存泄漏等问题。

4，总体方案

4.1 系统后台结构

● 服务器软件系统结构设计

服务器软件系统由界面显示模块、后台处理模块以及数据存储模块组成。

界面显示模块根据功能对服务器软件进行界面划分，包括实时在线状态显示、故障报警及查询、历史状态查询、故障统计以及维修信息的添加及查询。

后台处理模块主要完成与智能终端及界面的交互功能，包括以下几点：智能

终端的数据接收和处理，各种信息的存储、实时状态信息的更新、故障报警、历史信息的查询。

数据存储模块根据存储的数据类型进行划分，包括状态信息、故障信息以及维修信息。

服务器软件的系统结构如下图 2 所示：

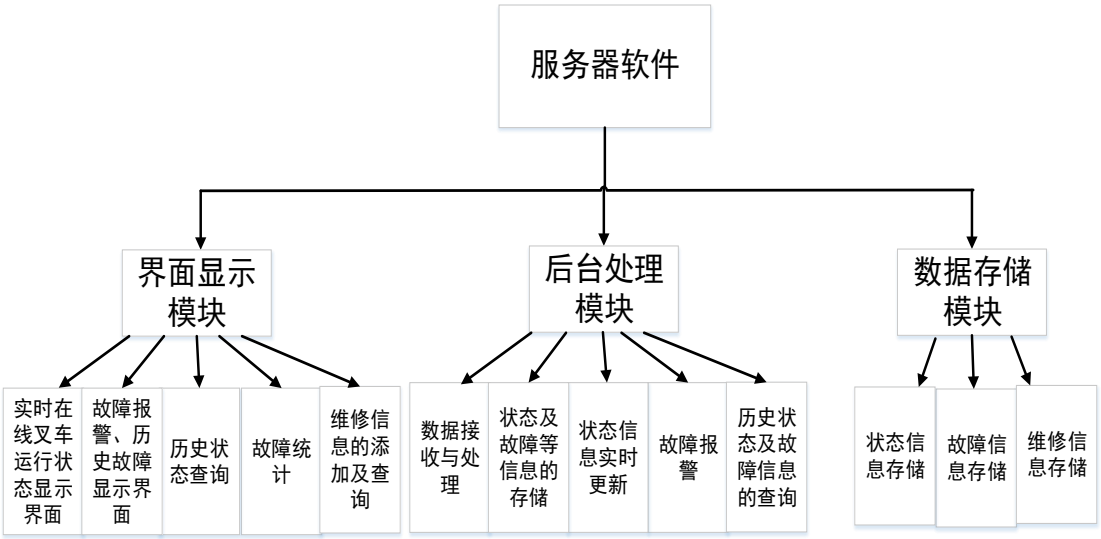

图 2 服务器软件架构

- 单机诊断软件系统结构设计

单机诊断软件系统由界面显示模块、后台处理模块组成。

其中界面显示模块按单机诊断软件功能进行划分，包括运行状态显示及故障报警显示。

后台处理模块通过串口与智能终端进行交互，完成数据接收与处理，状态信息实时更新、故障报警的功能。

单机诊断软件系统结构设计及子系统划分如下图 3 所示：

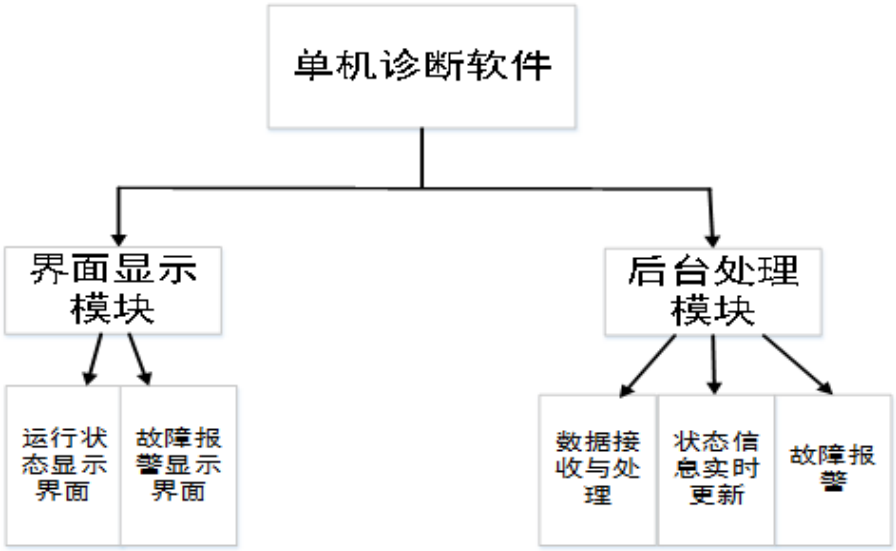

图 3 单机诊断软件架构

- 局域中心软件系统结构设计

局域中心软件系统由界面显示模块、后台处理模块组成。

其中界面显示模块按单机诊断软件功能进行划分，包括运行状态显示及故障报警显示。

后台处理模块通过 GPRS 或 WIFI 与智能终端进行交互，完成数据接收与处理，状态信息实时更新、故障报警的功能。

局域中心软件系统结构设计及子系统划分如下图4所示：

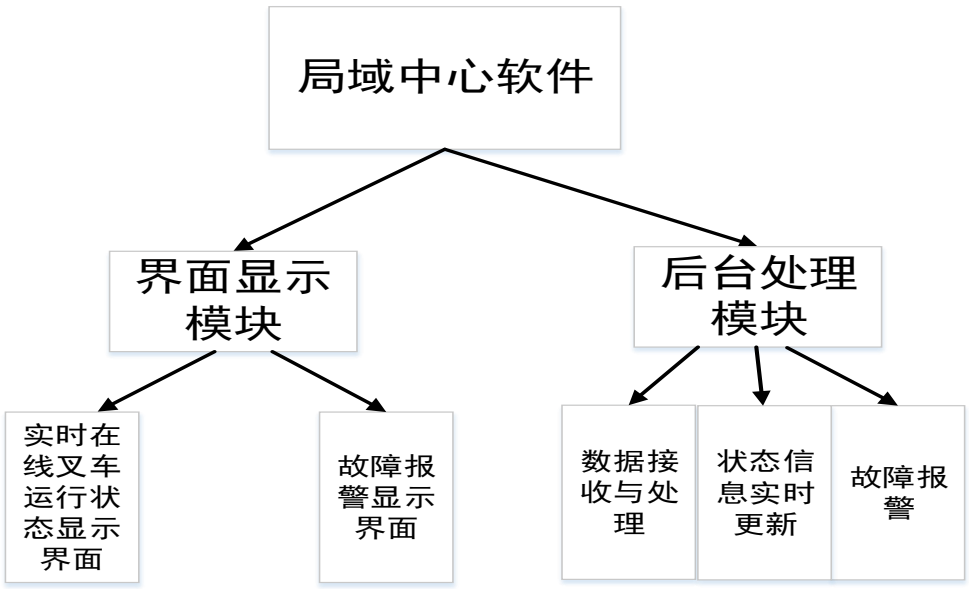

图 4 局域中心软件架构

4.2程序后台线程及数据流向

服务器软件采用多线程进行开发，后台逻辑较为复杂，特此做以下说明。

- 服务器软件后台程序线程说明

  服务器软件后台程序包含以下线程，共同协作满足各项功能需求：

  - ✧ BeginListen 线程，该线程建立一个 tcp 服务器，完成数据接收的功能，并把接收的数据存放在 Acceptdata 的队列容器中，等待后续线程进行数据解析；另外该线程将待查询的智能终端 IP 信息放入 ip2location 队列中，等待后续线程进行数据库 IP 地理位置查询。
  - ✧ dataprocess 线程，该线程完成数据处理解析的功能。获取 Acceptdata 队列中的数据进行解析，并根据功能码的不同对数据进行不同的处理方式（在线状态更新或故障检测）。
  - ✧ savaData 线程，获取 onlinedevicedataqueue 队列中的待存储数据，该线程完成数据的存储功能，将状态信息、故障信息存入数据库。
  - ✧ updateDeviceInfo 线程，获取 onlinedevicedataqueue 中的数据，对内存中叉车实时状态信息进行更新。

- 服务器软件数据流向说明

服务器后台主要包含三种类型数据：状态信息、故障报警信息、IP 地理位置数据，以下对上述三种数据的流向进行分别说明。

- ✧ 状态信息。流向为 beginlisten 线程->dataprocess 线程->savedata 线程存储到数据库或 updatedeviceinfo 线程更新实时状态信息。
- ✧ 故障信息。流向为 beginlisten 线程->dataprocess 线程->故障报警及存储
- ✧ IP 地理位置数据。流向为 beginlisten 线程->parseip 线程->更新叉车地理信息。

服务器软件后台线程及数据流向如下图 5 所示。

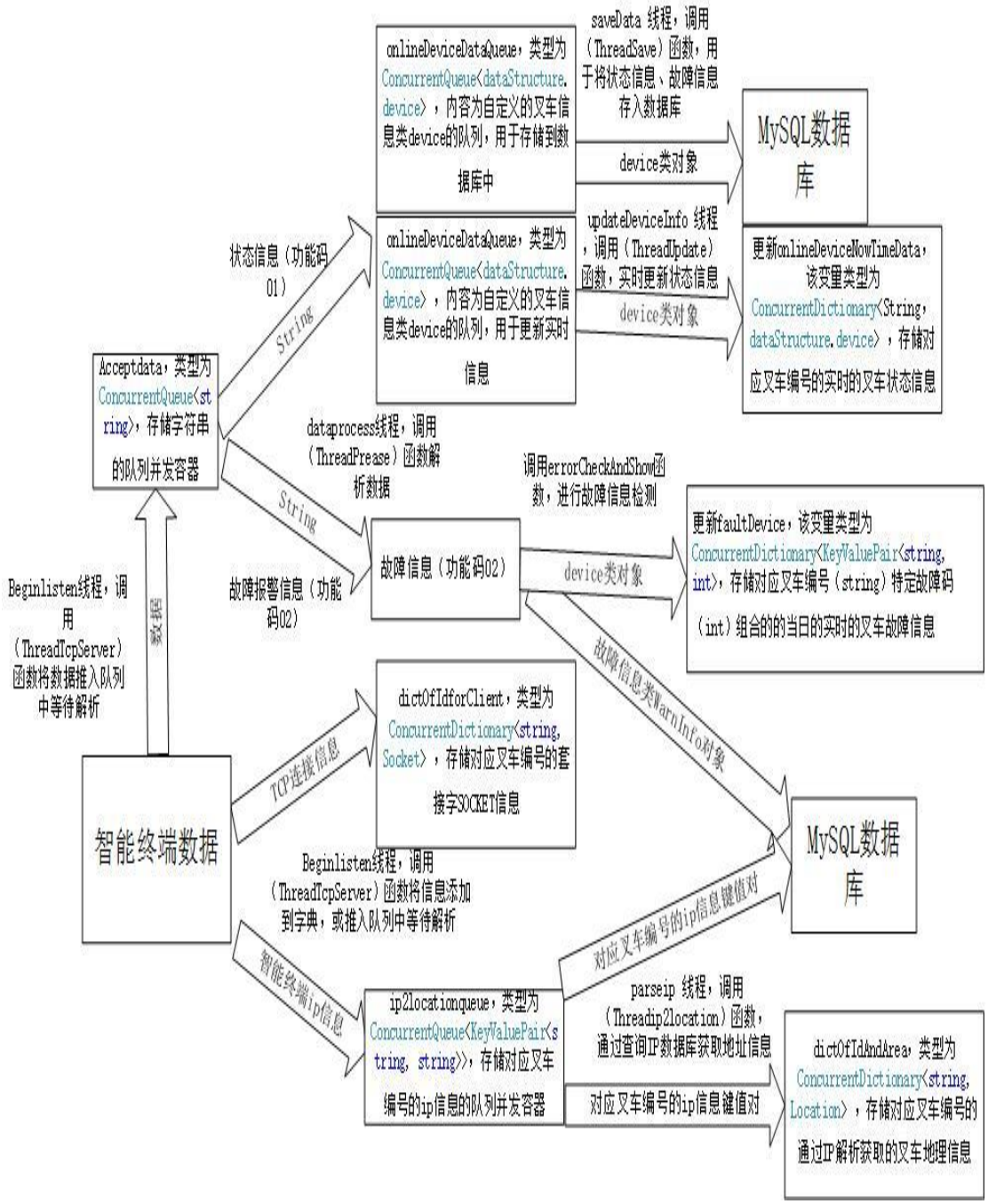

图 5 服务器软件后台线程与数据流

4.3通信协议

表 1

1字节	4字节	4字节	10字节	2字节	2字节	N字节	2字节	1字节
@	ADR	SRC	ID	FC	LEN	DATA	RC	$
帧头	目的地址	源地址	叉车ID	功能码	帧长	数据	校验	帧尾

其中数据部分内容如下：

表 2

10 字节	17 位	2 个	9 个	12 字节
运行时间	开关量	模拟量	CAN 数据	信息采集时间

功能码介绍如下：

表 3

功能码	数值	说明
FC_DAT	0x01	系统状态信息
FC_WNG	0x02	系统报警信息

4.4变量定义

在监控软件中定义了一些全局变量，用于在保存实时的数据信息，如在线叉车的实时状态信息等。下表是整个中心软件使用到的全局变量的类型以及其代表的含义。

表 4

变量名	类型	含义	定义位置
parseip	Thread	负责解析ip信息的线程	main.cs
saveData	Thread	负责将数据保存到数据库的线程	main.cs
dataprocess	Thread	负责数据处理的线程	main.cs
updateDeviceInfo	Thread	负责更新状态信息的线程	main.cs
beginlisten	Thread	负责开启TCP服务器，进行监听的线程	main.cs
server	Socket	服务器套接字	main.cs
dictOfIdforClient	ConcurrentDictionary<string, Socket>	叉车ID、客户端连接套接字字典	main.cs
dictOfIdAndArea	ConcurrentDictionary<string,	叉车ID、地区信息字典	main.cs

	Location>		
acceptdata	ConcurrentQueue<string>	接收数据（字符串）的队列	main.cs、standlone.cs
onlineDeviceDataQueue	ConcurrentQueue<dataStructure.device>	在线设备状态更新（添加）队列	main.cs、standlone.cs
onlineDeviceDatabaseQueue	ConcurrentQueue<dataStructure.device>	在线设备状态更新数据库（添加）队列	main.cs
ip2locationqueue	ConcurrentQueue<KeyValuePair<string,string>>	叉车ID、ip地址组成的pair值的待查询队列	main.cs
deviceStateSaveQueue	ConcurrentQueue<dataStructure.device>	设备状态储存队列	main.cs
errorDeviceDataQueue	ConcurrentQueue<dataStructure.WarnInfo>	设备故障或（解除故障）储存队列	main.cs
onlineDeviceNowTimeData	ConcurrentDictionary<string, dataStructure.device>	<叉车ID、实时信息对象>在线设备列表字典	main.cs、standlone.cs
buffer	ConcurrentDictionary<string, string>	<IP、缓存字符串>字典	main.cs
faultDevice	ConcurrentDictionary<KeyValuePair<string, int>, dataStructure.faultWarnData>	以<ID,故障码>作为key，具体故障信息对象为value故障报警字典	main.cs、standlone.cs
warnhistory	List<dataStructure.faultWarnData>	历史故障信息缓存列表	main.cs
language	string	保存软件界面选择的语言（中文、english）	main.cs
selectStateName	string	保存上次选择的状态信息节点的名字（历史状态信息查询中使用）	main.cs
selectNodeText	string	保存上次选择的状	main.cs

		态信息节点的文本信息（历史状态信息查询中使用）	
istoday	int	判断是否过了一天，用于进入新的一天时将内存中故障信息清零	main.cs
mainformhandle	Main	主窗口类句柄，用于显示实时数据	deviceInfo.cs、repair.cs
Stated	string	叉车ID	deviceInfo.cs
Language	string	保存软件界面选择的语言（中文、english）	deviceInfo.cs
Standloneid	string	叉车ID（单机诊断软件）	standalone.cs
standlonerunningtime	string	叉车运行时间（单机诊断软件）	standalone.cs
_serialPort	SerialPort	串口资源类对象	standalone.cs
_continue	bool	串口是否打开标志	standalone.cs

4.5数据类定义

1） WarnInfo类（dataStructure.cs）

表5

成员	含义
Id	报警信息的叉车 ID
Time	报警时间
errorValue	报警值
errorNumber	故障码
errorLevel	故障等级
errorState	故障状态（故障状态为0时，故障开始报警，故障状态为1时，故障正在进行中，为2时故障警报解除）

2） faultWarnData类（dataStructure.cs）

表6

成员	含义
faultID	报警信息的叉车 ID
faultTime	报警时间的列表，与报警值、报警状态一一对应
errorValue	报警值的列表，与报警时间、报警状态一一对应
errorState	故障状态的列表（故障状态为0时，故障开始报警，故障状态为1时，故障正在进行中，为2时故障警报解除），与报警时间、报警值一一对应
errorNumber	故障码
errorLevel	故障等级
errorCount	报警次数

3） repairData类（dataStructure.cs）

表7

成员	含义
repairID	维修养护信息的叉车 ID
repairTime	维修养护时间
repairman	维修人员工号
Remark	备注
repairRecord	维修内容

4） device类（dataStructure.cs）

表8

成员	含义	对应字符串 data 数据部分位置
Id	叉车 ID	不属于数据部分，直接字符串传入赋值
runningTime	叉车运行时间	0-9
Time	信息采集时间	73-84
hornSwitch	喇叭正极	10
Horn	喇叭开关	（11-12 位）&0x80
hornCount	喇叭动作次数	（11-12 位）&0x7f
hornGround	喇叭负极	27
upBtnSwitch	提升开关正极	13
upBtnCount	提升开关动作次数	（14-15 位）&0x80
upBtnContactorCoilUpper	提升接触器线圈进线端	（14-15 位）&0x7f
upBtnContactorCoilDo	提升接触器线圈接地端	28

wn		
upBtnContactorCount	提升接触器动作次数	（17-18 位）&0x7f
upBtnContactUpper	提升触点进线端	16
upBtnContactDown	提升触点出线端	（17-18 位）&0x80
downBtnSwitch	下降开关正极	19
downBtnCount	下降开关动作次数	（20-21 位）&0x7f
downBtnsolenoidvalveUpper	下降电磁阀进线端	（20-21 位）&0x80
downBtnsolenoidvalveDown	下降电磁阀接地端	29
masterContactorCoilUpper	主接触器线圈进线端	22
masterContactorCoilDown	主接触器线圈接地端	30
masterContactUpper	主接触器触点进线端	23
masterContactDown	主接触器触点出线端	（24-25 位）&0x80
masterContactorCount	主接触器动作次数	（24-25 位）&0x7f
arresterUpper	制动器线圈控制端	26
arresterDown	制动器线圈正极	（31-32 位）&0x80
arresterCount	制动器动作次数	（31-32 位）&0x7f
liftMotorCurrent	提升电机电流	33-34
liftMotorTemperature	提升电机温度	35-36
canDirectionandSpeedMode	行走控制器运行方向及高低速模式选择、紧急反向以及 interlock	37-38 分别为该值与某个数取与的结果 紧急转向： canDirectionandSpeedMode & 0x0004 Interlock: canDirectionandSpeedMode & 0x0010 运行方向： canDirectionandSpeedMode & 0x0003 高低速模式： canDirectionandSpeedMode & 0x0008
canSpeed	行走控制器转速	39-42
canError	故障码	43-44 int 数据最高位为故障标志位 canError&0x0080>0 代表

		故障报警； 否则代表故障恢复
canLowPowerMode	行走控制器低功耗模式	45-46
canCourse	行走控制器小计里程	47-50
canDirectVoltage	行走控制器电机电压	55-58
canMotorCurrent	行走控制器电机电流	59-62
canMotorTemperature	行走控制器电机温度	63-66
isCanConnection	与行走控制器通信是否正常	53-54 71-72 两者均为 ff 时代表通信故障，否则代表通信正常

5） Location类（Location.cs）

表9

成员	含义
ip_from	ip 起始值
ip_to	ip 终止值
country_code	国家代码
country_name	国家名称
region_name	省份名称
city_name	城市名称

4.6系统界面设计

根据模块功能对系统界面进行划分，系统分为以下几个界面。

◆ 实时在线叉车运行状态显示界面

该界面中显示在线叉车的状态显示，具体内容包括：

✧ 区域分布树状图（根据地域信息按国家省市对叉车分类，方便查询）。

✧ 特定叉车查询区域(可以根据编号或者区域名称对叉车进行查询,显示满足特定条件的叉车)。

✧ 在线设备列表（显示满足 i 或 ii 条件的叉车列表，显示叉车基本信息，点击后可以查看叉车具体信息）。

✧ 日历、时间及搭配图片等，美化界面，方便用户。

实时在线叉车运行状态显示界面的设计如图 6 所示。

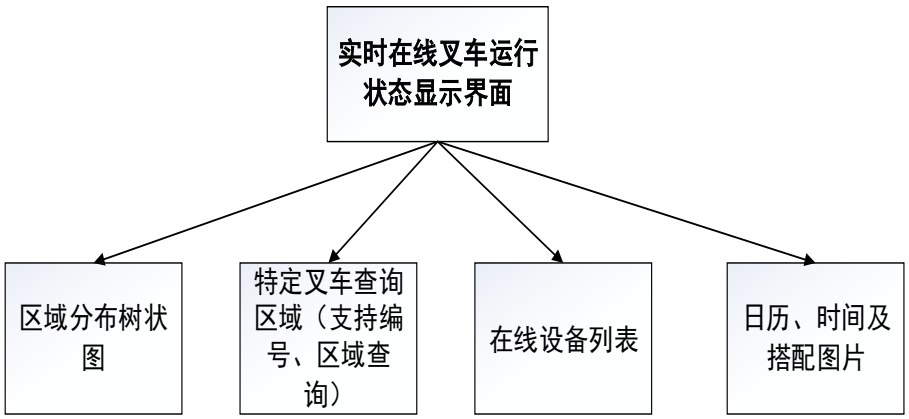

图6 实时在线叉车运行状态显示界面设计

◆ 叉车具体信息显示（在线叉车运行状态子界面）

该界面包含以下内容：

✧ 叉车基本信息（编号、运行时长等）。

✧ 行走控制器信息（数据来源为控制器 can 数据，包含电压、电流、温度、转速等）。

✧ 提升控制器信息（数据来源为智能终端采集的关于提升控制器的信息）。

✧ 喇叭等信息（数据来源为智能终端采集的非提升控制器部分的信息）。

叉车具体信息显示设计如图 7 所示。

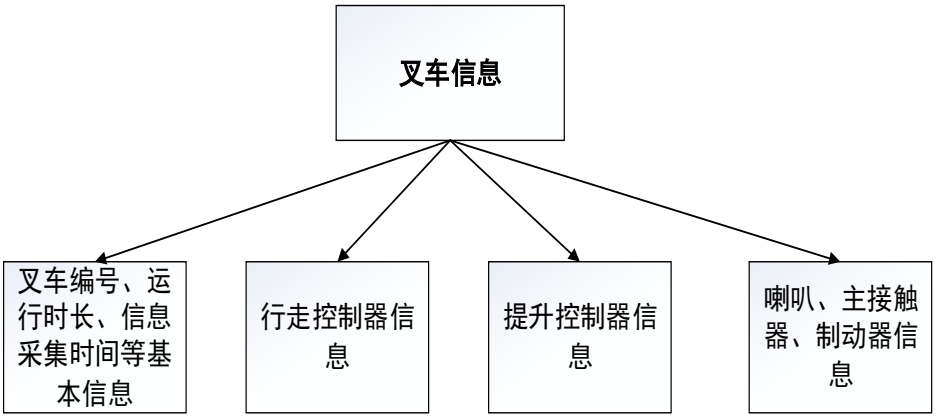

图7 叉车具体信息界面设计

◆ 故障报警、历史故障显示界面

该界面中显示故障报警、历史故障，具体内容包括：

✧ 今日故障查询区域（可以根据叉车编号或者故障码对故障进行查询）。

✧ 今日故障显示列表（显示满足查询条件今日故障列表，故障按叉车编号与故障码的组合进行罗列）。

✧ 历史故障查询区域（可以根据叉车编号对故障进行查询）。

✧ 今日故障显示列表（显示满足查询条件的历史故障列表，故障按故障码的顺序进行罗列）。

故障报警、历史故障显示界面设计如图 8 所示。

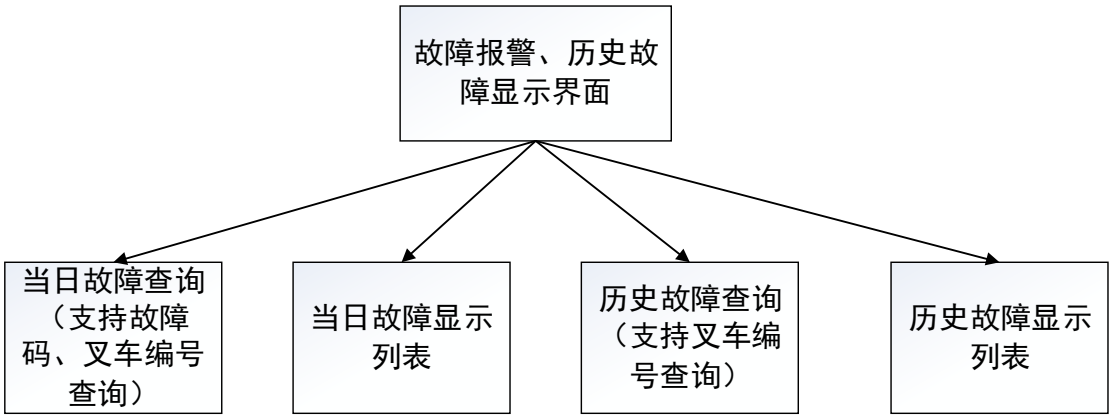

图8 故障报警、历史故障显示界面设计

◆ 故障报警具体信息显示（故障报警、历史故障子界面）

该界面包含以下内容：

✧ 叉车基本信息（编号）。

✧ 故障代码、发生次数。

✧ 故障信息名称、故障处理方法。

✧ 开始报警、最近报警时间、故障报警时间列表。

故障报警具体信息显示设计如图 9 所示。

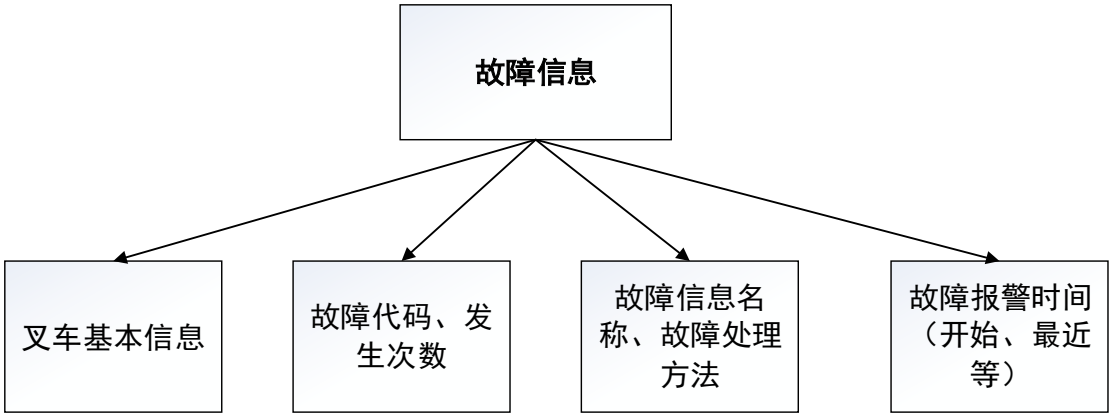

图9 故障报警具体信息显示界面设计

◆ 历史状态查询界面

该界面中显示历史状态信息，具体内容包括：

✧ 叉车信息查询项树状图（选取特定的查询项）。

✧ 查询选择区域（选取需要查询的编号、时间范围）。

✧ 模拟量显示（以图表曲线的形式显示模拟量的变化）。

✧ 数字量显示（显示一定时间范围内的数字量（开关等）变化次数）。

历史状态查询界面设计如图 10 所示。

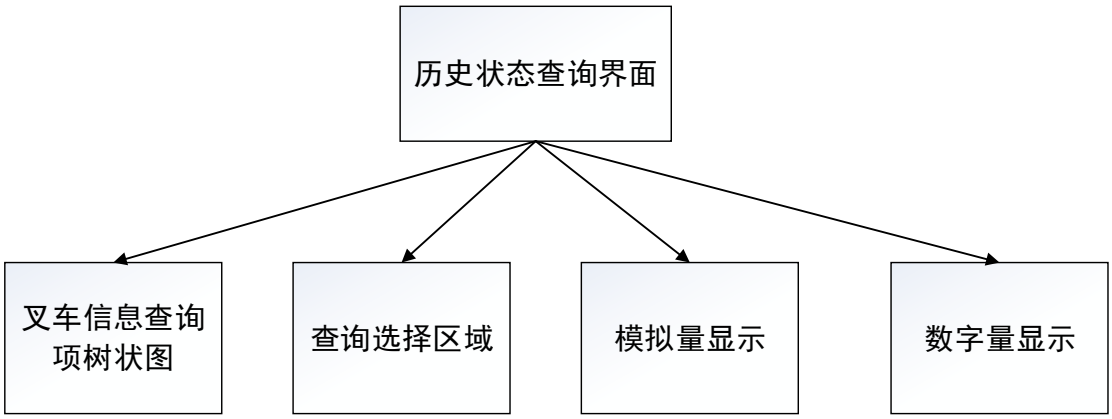

图10 历史状态查询界面设计

◆ 维修信息的添加及查询界面

该界面中显示维修信息的添加及查询信息，具体内容包括：

✧ 维修信息添加及查询

✧ 历史维修信息列表。

维修信息的添加及查询界面设计如图 11 所示。

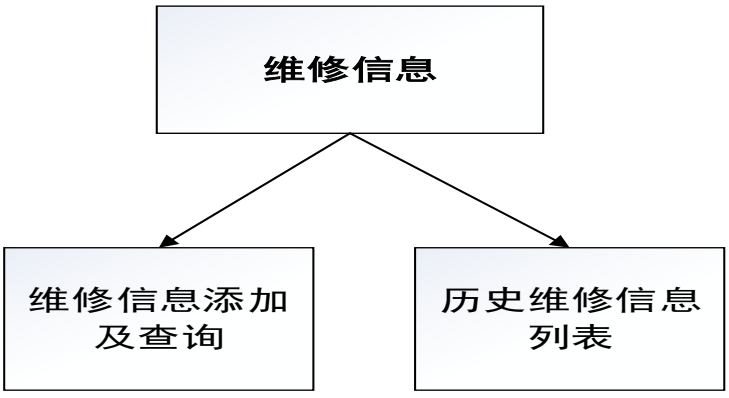

图11 维修信息的添加及查询界面设计

◆ 维修保养清单添加界面

包含以下内容：维修保养的叉车编号，维修人员、维修内容、维修时间。

维修保养清单添加界面设计如图 12 所示。

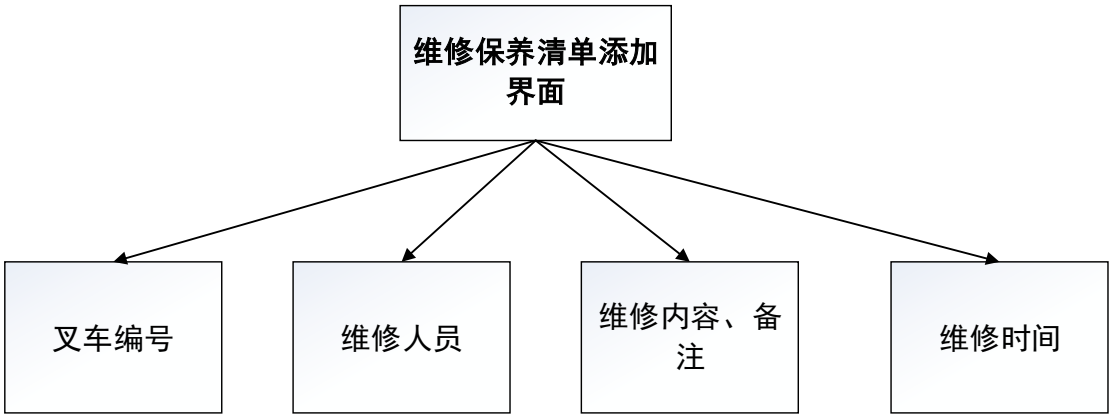

图12 维修保养清单添加界面设计

◆ 故障统计界面

该界面中显示故障统计信息，具体内容包括：

✧ 单辆叉车故障分布信息统计。

✧ 一定生产日期内叉车故障分布信息统计。

✧ 所有叉车故障分布信息统计。

故障统计界面设计如图 13 所示。

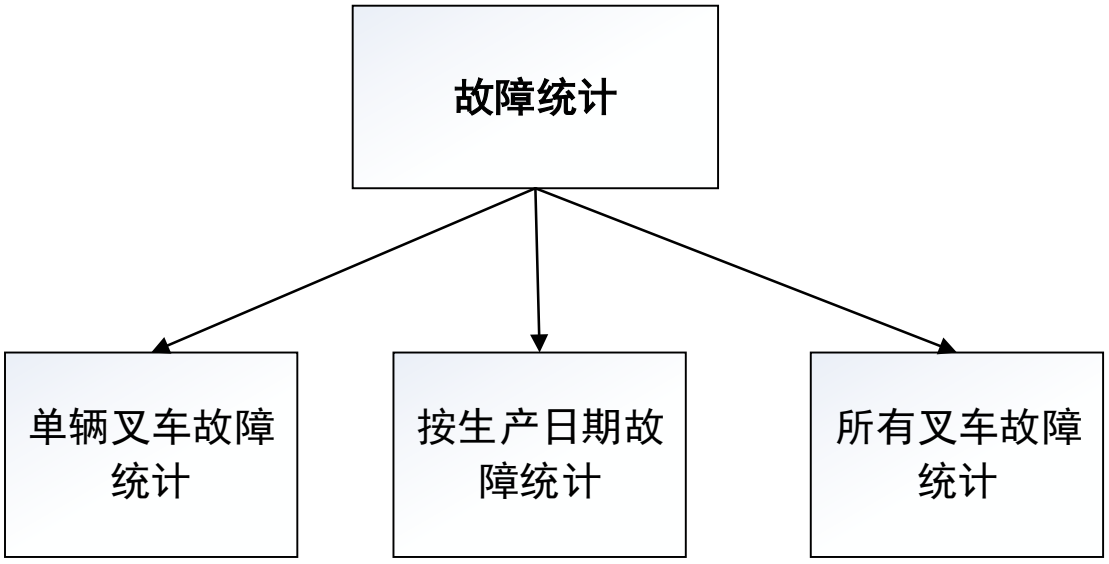

图13 故障统计界面设计

5，模块详细设计

5.1程序入口

整个程序的入口函数，在该函数中绑定线程调用函数，并启动以下线程：

- ✧ 解析 ip 线程：该线程通过 ip 查询地域信息数据库，获取叉车的地域信息。
- ✧ 储存线程：存储获取到的智能终端数据。
- ✧ 在线状态更新线程：更新在线叉车的状态信息。
- ✧ 数据处理线程：数据处理，对数据进行解析。
- ✧ 数据接收线程：接收来自智能终端的数据信息。

数据流向为数据接收->数据处理->在线状态更新、IP 解析、数据存储线程，因此线程启动的顺序应该为在线状态更新、IP 解析、数据存储线程->数据处理线程->数据接收线程。

相关源码如下：

public main(string lang) （main.cs）

```
{
    ……

    //声明以下四个线程，并设置其为后台工作，并启动
     Thread parseip = new Thread(Threadip2location);
     Thread saveData = new Thread(ThreadSave);
     Thread dataprocess = new Thread(ThreadPrease);
     Thread updateDeviceInfo = new Thread(ThreadUpdate);
     parseip.IsBackground = true;
     updateDeviceInfo.IsBackground = true;
     dataprocess.IsBackground = true;
     saveData.IsBackground = true;
     saveData.Start();
     parseip.Start();
     updateDeviceInfo.Start();
```

```
dataprocess.Start();
……

}
```

线程启动流程图如图 14 所示。

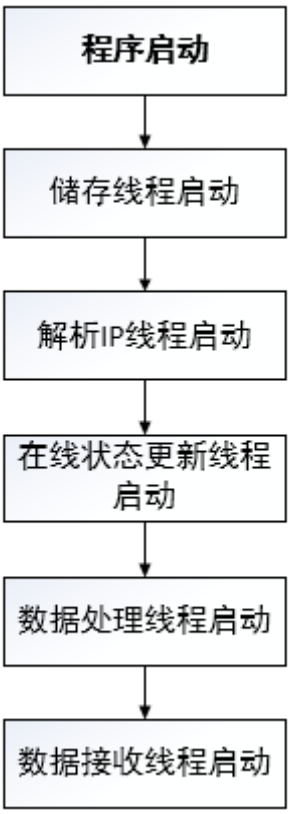

图 14 线程启动流程图

5.2数据接收模块

数据接收模块采用基于 select 的 IO 复用的服务器模型进行端口监听。在该模块中，首先对服务器端口进行绑定并开始监听套接字变化状态：

✧ 当服务器套接字发生变化时，代表有新的智能终端连接；

✧ 其他套接字发生，代表有数据要读写或发生其他变化（断开连接等）。

相关源码如下：

public void ThreadTcpServer(object port) （main.cs）

{

```
try {
IPAddress ipaddress = GetLocalIPV4(); //获取本机 ip 地址

IPEndPoint iep = new IPEndPoint(ipaddress, (int)port);

server = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp); //定义服务器对象

// 将套接字与本地终结点绑定
server.Bind(iep); //绑定 ip

server.Listen(5000);//开始监听

List<Socket> socketList = new List<Socket>(); //套接字列表
socketList.Add(server);//将服务端添加进来

while (true)
{

List<Socket> temp = socketList.ToList(); //复制临时列表
Socket.Select(temp, null, null, 1000); //筛选出发生变化的套接字
int count = temp.Count;

for (int i = 0; i < count; i++)
{
if (temp[i].Equals(server)) //发生变化的是服务端，意味着有新的连接
{
Socket client = socketList[i].Accept(); //建立新的客户端连接
socketList.Add(client);//加入套接字列表
```

```
}
else //发生变化的是客户端，代表着数据需要接收
{
byte[] bytes = new byte[1024]; //缓存数组
 if ((len = temp[i].Receive(bytes)) > 0)
……
}
```

数据接收模块流程图如图 15 所示。

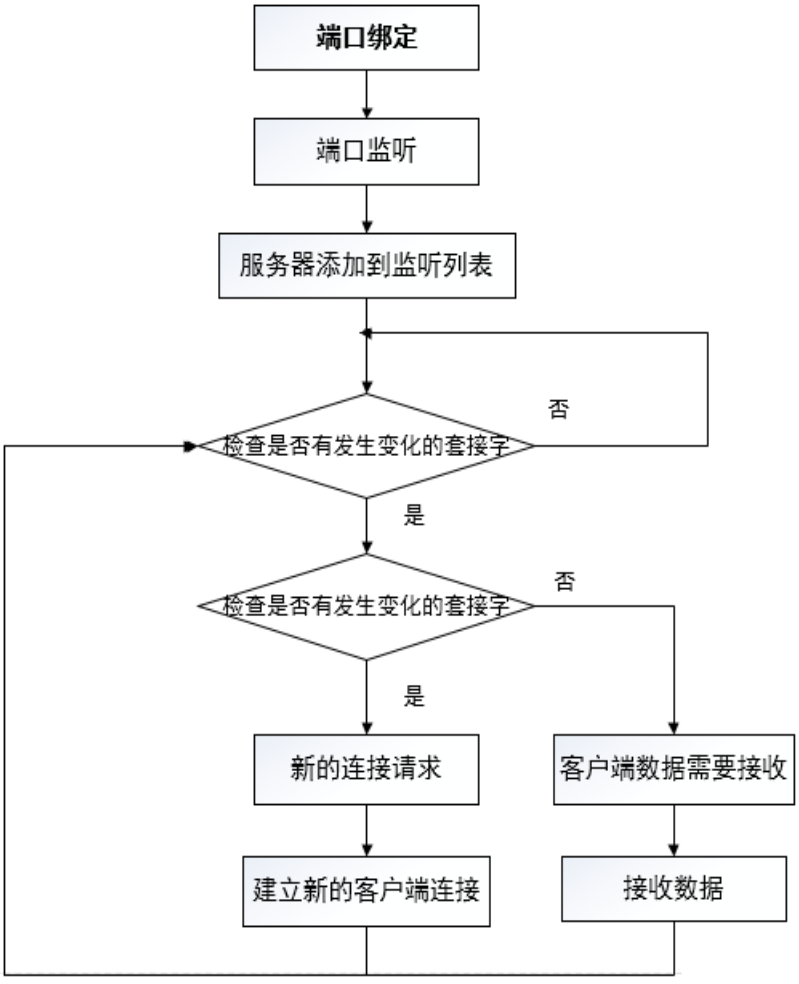

图 15 数据接收模块流程图

5.3数据处理模块

数据处理模块对接收到智能终端信息，根据功能码不同分别进行解析：

✧ 状态信息，更新在线设备的状态，同时进行数据存储。

✧ 故障信息，检查状态信息中是否有故障，进行故障报警与存储。

相关源码如下：

private void dataprocess(string data) （main.cs）

```
        {
            ……
            string fc = data.Substring(dataHead + 19, 2);//获取功能码

        ……
            switch (fc) //根据功能码不同分别进行解析
            {
                //状态采集信息处理
                case "01":

                    dataStructure.device deviceState = new dataStructure.device(id, collectData);
                    //推入在线设备更新队列、设备状态信息存储队列
                    onlineDeviceDataQueue.Enqueue(deviceState);
                    deviceStateSaveQueue.Enqueue(deviceState);
                    //onlineDeviceDatabaseQueue.Enqueue(deviceState);

                    break;
                case"02":
                    dataStructure.device FaultdeviceState = new dataStructure.device(id, collectData);

                    ///检查状态信息中是否有故障
                    errorCheckAndShow(ref faultDevice, FaultdeviceState);
                    break;
                default:
                    break;
```

}

数据处理模块流程图如图 16 所示。

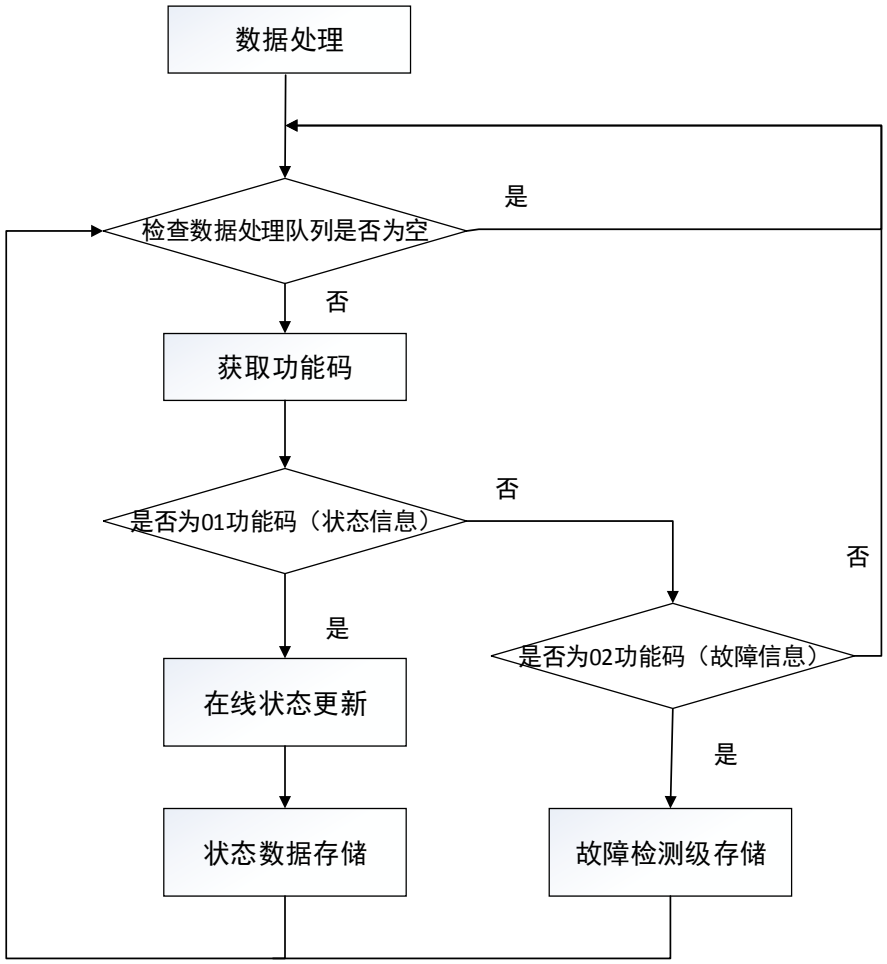

图 16 数据处理模块流程图

5.4故障检测模块

故障检测模块，首先判断故障信息高位状态：

- ✧ 高位为 1 代表有故障，检测故障类型，存储故障信息，并更新界面进行报警；
- ✧ 高位为 0 代表故障恢复，存储故障恢复信息，更新界面恢复对应故障。

相关源码如下：

public void errorCheckAndShow(ref ConcurrentDictionary<KeyValuePair<string,

int>, dataStructure.faultWarnData> faultDevice, dataStructure.device deviceState)（main.cs）

{

if ((deviceState.canError&0x0080)>0) // 如果有故障发生 canError 高位为 1 代表有故障

{

……

errorDeviceDataQueue.Enqueue(save);//存储故障信息

this.Invoke((Action)(……）//更新界面报警信息

}

else //高位不为零，代表故障恢复

{

……

errorDeviceDataQueue.Enqueue(save);

this.Invoke((Action)(……）

}

}

故障检测模块流程图如图 17 所示。

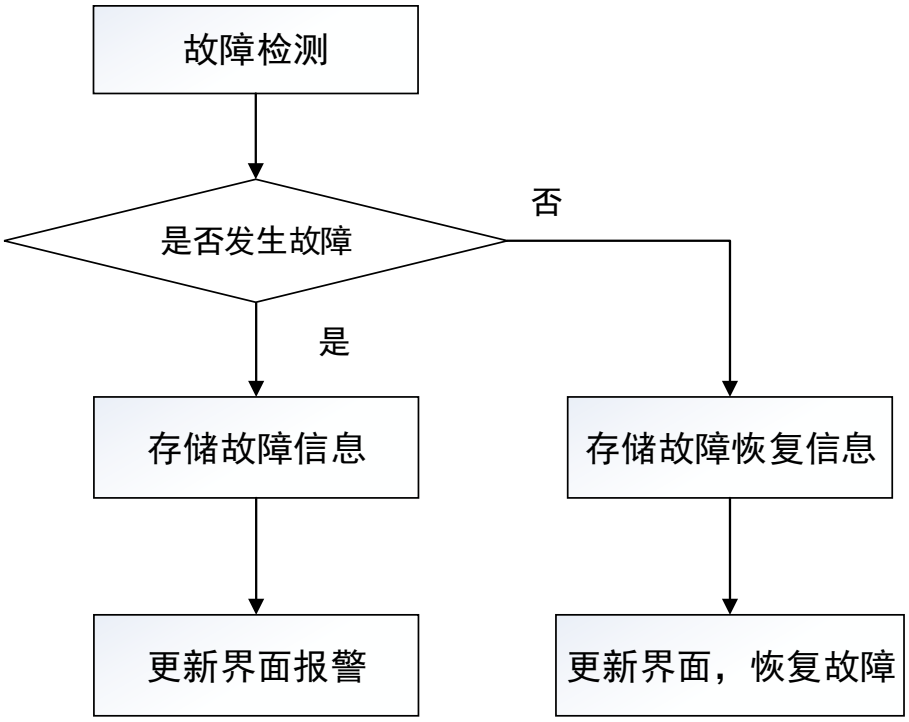

图 17 故障检测模块流程图

5.5数据存储模块

储存线程模块，对状态信息和故障信息进行存储。为了增快存储速度，采用 MySQL 的事务进行存储。首先检查存储队列是否为空，不为空的话，将存储的数据拼接成 sql 语句，打开数据库连接，按 500 个一次进行事务提交，提交完毕后关闭数据库。

相关代码如下：

```
public void ThreadSave()（main.cs）
{
while (true)
{
try
{
if (!deviceStateSaveQueue.IsEmpty)  //在线状态储存队列是否为空，不空的话进行储存
{
List<string> SQLStringList = new List<string>();  //需要存储的事务列表
```

```
int m = 0;
while(!deviceStateSaveQueue.IsEmpty&&m<1000)  //需要存储的队列已为空或者已经有 1000 条需要存储的信息
{
dataStructure.device device;
deviceStateSaveQueue.TryDequeue(out device); //取出字符串元素
string sql = ……
SQLStringList.Add(sql); //添加到事务列表中

}

//打开数据库连接，进行存储
using (MySqlConnection thisConnection = new MySqlConnection("Database=ruyi;Data Source=localhost;User ID=root;password=123456"))
{
try { thisConnection.Open();
MySqlCommand cmd = new MySqlCommand();
cmd.Connection = thisConnection;
MySqlTransaction tx = thisConnection.BeginTransaction();
cmd.Transaction = tx;
try
{
for (int n = 0; n < SQLStringList.Count; n++)  //遍历列表中元素
{
string strsql = SQLStringList[n].ToString();
if (strsql.Trim().Length > 1)          //加入事务中
{
cmd.CommandText = strsql;
```

```
cmd.ExecuteNonQuery();
}
//后来加上的
if (n >= 0 && (n % 500 == 0 || n == SQLStringList.Count - 1)) //500 个或虽然没有到 500，但包含所有的话就提交事务
{
tx.Commit();
tx = thisConnection.BeginTransaction();
}
}
//tx.Commit();//原来一次性提交
}
catch (System.Data.SqlClient.SqlException E)
{
// tx.Rollback();
throw new Exception(E.Message);
}
}catch(Exception e)
{
thisConnection.Close();
}
}

……
}
```

数据处理模块流程图如图 18 所示。

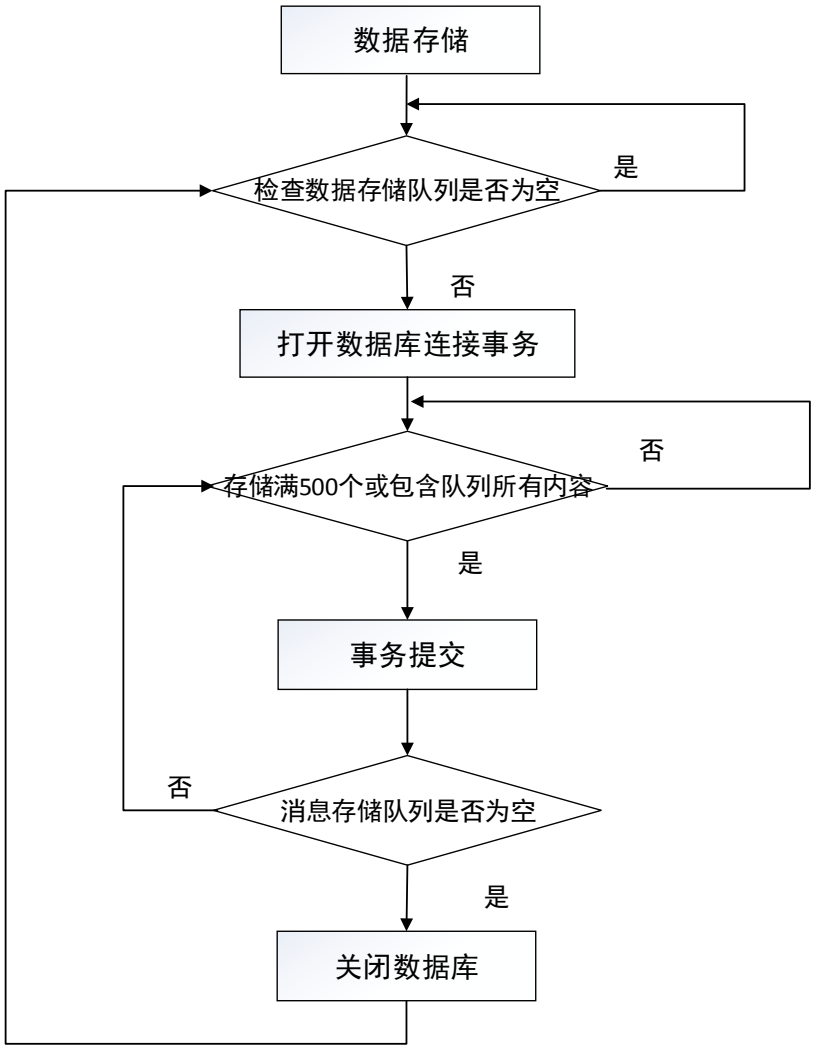

图 18 数据存储模块流程图

6，非功能部分设计

6.1 编程规范

采用以下编程规范，保证程序稳定性。

- 采用 c#进行编程，该语言有智能回收功能，能有效避免内存泄漏。
- 所有程序体都用 try……catch 进行包裹，能够有效捕获异常，避免程序崩溃。
- 规范代码，调用对象之前检查是否为空指针，或者是否容器内元素为空。

6.2 软件性能及稳定性设计

- 对软件进行了压力测试，模拟 1 万台智能终端向监控软件发送数据，检测是否能正常运行。
- 模拟数据库断开连接等故障，检测对软件的影响。
- 使用各种类型的误操作，检测是否会使软件崩溃。

7，数据库系统设计

7.1 设计要求

设计的数据库系统可以存储 1 万台智能终端的状态及故障数据，在不发生断网故障的情况下能够稳定运行，长期运行后增删改查速度较快（ms 级别）。

7.2 数据库设计

7.2.1 设计依据

由于监控软件每分钟会接收到智能终端的状态信息并进行存储，因此状态信息存储量较大，不适合放在一张表里；否则数据较多时会导致查询及插入速度很慢。因此对状态信息按叉车编号进行取余分表，分为 100 张表；同时对故障信息也进行取余分表，分为 10 张表。维修信息较少，放在一张表即可。叉车区域分布表存储根据 ip 数据库查询到的叉车区域分布信息。

7.2.2 数据库种类及特点

选用 MySQL 作为存储数据库。MySQL 属于关系型数据库，在企业中应用广泛，且轻量级，使用方便。

7.2.3状态信息结构设计

将状态信息进行取余分表，将叉车的 ID 对 100 取余，放入对应表中。例如 201602240001%100=1，所以放在 deviceinfo_1 中；201602240028%100=28，所以放在 deviceinfo_28 中。各字段的数据类型与含义如下表所示。

表 10

字段名称	数据类型	含义	是否主键
Id	Char	叉车 ID	是

runningTime	Datetime	叉车运行时间	否
Time	Tinytext	信息采集时间	是
hornSwitch	Tinyint	喇叭正极	否
Horn	Tinyint	喇叭开关	否
hornCount	Int	喇叭动作次数	否
hornGround	Tinyint	喇叭负极	否
upBtnSwitch	Tinyint	提升开关正极	否
upBtnCount	Int	提升开关动作次数	否
upBtnContactor CoilUpper	Tinyint	提升接触器线圈进线端	否
upBtnContactor CoilDown	Tinyint	提升接触器线圈接地端	否
upBtnContactor Count	Int	提升接触器动作次数	否
upBtnContactUp per	Tinyint	提升触点进线端	否
upBtnContactDo wn	Tinyint	提升触点出线端	否
downBtnSwitch	Tinyint	下降开关正极	否
downBtnCount	Int	下降开关动作次数	否
downBtnsolenoi dvalveUpper	Tinyint	下降电磁阀进线端	否
downBtnsolenoi dvalveDown	Tinyint	下降电磁阀接地端	否
masterContacto rCoilUpper	Tinyint	主接触器线圈进线端	否
masterContacto rCoilDown	Tinyint	主接触器线圈接地端	否
masterContactU pper	Tinyint	主接触器触点进线端	否
masterContactD own	Tinyint	主接触器触点出线端	否
masterContacto rCount	Int	主接触器动作次数	否
arresterUpper	Tinyint	制动器线圈控制端	否
arresterDown	Tinyint	制动器线圈正极	否
arresterCount	Int	制动器动作次数	否
liftMotorCurre nt	Int	提升电机电流	否
liftMotorTempe rature	Int	提升电机温度	否
canDirectionan dSpeedMode	Int	行走控制器运行方向及高低速模式选择、	否

		紧急反向以及 interlock	
canSpeed	Int	行走控制器转速	否
canError	Int	故障码	否
canLowPowerMode	Int	行走控制器低功耗模式	否
canCourse	Int	行走控制器小计里程	否
canDirectVoltage	Int	行走控制器电机电压	否
canMotorCurrent	Int	行走控制器电机电流	否
canMotorTemperature	Int	行走控制器电机温度	否

7.2.4 故障信息结构设计

将故障信息进行取余分表，将叉车的 ID 对 10 取余，放入对应表中。例如 201602240001%10=1，所以放在 warninfo_1 中；201602240028%10=8，所以放在 warninfo_8 中。各字段的数据类型与含义如下表所示。

表 11

字段名称	数据类型	含义	是否主键
Id	Char	叉车 ID	是
errorNumber	Int	故障码	是
errorTime	Datetime	故障发生时间	是
errorValue	Int	故障报警值	否
errorLevel	Tinyint	故障等级	否
errorstate	Tinyint	故障状态	是

7.2.5 维修信息结构设计

维修记录表记录维修的相关信息。各字段的数据类型与含义如下表所示。

表 12

字段名称	数据类型	含义	是否主键
Id	Char	叉车 ID	是
repairTime	Datetime	维修时间	是
repairMan	Varchar	维修人员工号	否
remark	Text	备注	否

repairRecord	Text	维修记录	否

7.2.6区域分布信息结构设计

根据叉车智能终端的 IP 信息，对叉车进行区域划分，将结果存入到该表中。各字段的数据类型与含义如下表所示。

表 13

字段名称	数据类型	含义	是否主键
Id	Char	叉车 ID	是
ip	Datetime	叉车 IP 值	否
country_code	Char	国家代码	否
country_name	Varchar	国家名称	否
region_name	Varchar	省份名称	否
city_name	Varchar	城市名称	否

7.2.7使用的数据库——ip地域信息数据库

原数据库为 ip2location_db3，覆盖所有 ip 地址，因为数据较多（400w 行），分为 ip2location_1—ip2location_9 共 10 张表，分别存储部分 ip 区域信息。查询时将 IP 地址转换为整形，根据值的大小查询对应的数据表。各字段的数据类型与含义如下表所示。

表 14

字段名称	数据类型	含义	是否主键
Id	Char	叉车 ID	是
ip	Datetime	叉车 IP 值	否
country_code	Char	国家代码	否
country_name	Varchar	国家名称	否
region_name	Varchar	省份名称	否
city_name	Varchar	城市名称	否