天津中德应用技术大学

Tianjin Sino-German University of Applied Sciences

本科生毕业设计

定制化加工生产线的 MES 系统设计

Design of MES system for Customized Production Line

姓　　名　王天乐

学　　院　智能制造学院

专　　业　自动化

指导教师　史艳霞

职　　称　教授

完成时间　2021.06.04

天津中德应用技术大学

本科生毕业设计（论文）的声明

本人郑重声明：所呈交的毕业设计（论文），是本人在指导教师指导下，进行研究工作所取得的成果。除文中已经注明引用的内容外，本毕业设计（论文）的研究成果不包含任何他人创作的、已公开发表或没有公开发表的作品内容。对本设计（论文）所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。本毕业设计（论文）原创性声明的法律责任由本人承担。

毕业设计（论文）作者签名：

年　　月　　日

本人声明：该毕业设计（论文）是本人指导学生完成的研究成果，已经审阅过设计（论文）的全部内容，并能够保证题目、关键词、摘要部分中英文内容的一致性和准确性。

毕业设计（论文）指导教师签名：

年　　月　　日

摘　要

MES 系统是工厂、设备、客户之间的桥梁，在产品计划到成品产出的整个过程扮演生产活动最佳化的“信息传递者”的角色。工厂的决策信息和客户的订单信息通过 MES 系统传达给设备，MES 系统根据订单的需要制定出包括产品类型、数量等的生产计划与生产步骤。同时，MES 系统可以对每个工位所安装的部件类型、作业方法以及原材料进行设置，以便对设备和材料进行实时监控。

本课题在定制化加工生产线的基础上，提出了数据库、网络通信、数据加密等技术相结合的定制化加工生产线 MES 系统设计。本课题的 MES 系统由数据库、MES 服务器、Web 页面、现场监控大屏程序等共 4 个部分组成，其中 MES 服务器、Web 页面、现场监控大屏程序运用了模块化设计，使得程序易于维护、修改和功能扩充。MES 服务器可直接读写数据库，Web 页面和现场监控大屏程序需要与 MES 服务器建立网络连接，通过向 MES 服务器发送请求来获取相关信息或执行相关操作。MES 服务器会对订单进行排序、执行与监控，MES 服务器的生产控制模块与生产线上的 PLC 程序相互配合，使得 MES 服务器可以对生产线上的各个站进行精确控制，从而控制产品的生产与订单的执行。使用了 MES 系统的定制化加工生产线具有以下特点：信息和指令的传达速度得到提升，订单及原材料管理更加便捷，生产效率更高、系统稳定性更好，产品可追踪可追溯，生产活动可见可控。

本课题既能体现工业 4.0 与中国制造 2025 中智能化和信息化的特点，又符合现代工业加工生产线对自动化和效率的要求，对传统的工业加工生产线改造升级有着一定的意义和影响。

关键词：制造执行系统；MES 系统；定制化加工生产线

ABSTRACT

Manufacturing Execution System (MES) is a bridge among factory, equipment and customers, and plays the role of information transmitter in the whole process from product planning to finished product output. The decision information of the factory and the order information of the customer are transmitted to the equipment through MES. MES formulates the production plan and production steps including product type and quantity according to the order. At the same time, MES can configure the components' type, operation method and raw materials installed in each unit, in order to monitor the equipment and materials in real time.

On the basis of the customized production line, this subject proposes the design of MES for the customized production line, which combines database, network communication, data en/decryption and other technologies. The MES of this subject is composed of four parts which are database, MES Server, Web Pages, and On-Screen-Monitoring Program. Modular design is used in MES server, web pages, and on-screen-monitoring program, making the program easy to be maintained, modified and expanded. MES Server can read and write database directly. Web Pages and On-Screen-Monitoring Program need to connect to MES Server through network, and then they are able to obtain information or perform operations by sending requests to MES Server. MES Server will sort, execute and monitor orders automatically. The production control module of MES Server cooperates with PLC program on the production line, so that MES Server can accurately control each station on the production line, so as to control the production of products and the execution of orders. The customized production line with MES used has the following characteristics: improve the communication speed of information and instructions; facilitate the management of orders and raw materials; higher production efficiency and better system stability; make products traceable, make production activities controllable.

This subject can not only reflect the characteristics of intelligentization and informatization in Industry 4.0 and Made in China 2025, but also meet the requirements of automation and efficiency of modern industrial production lines. It has a certain significance and influence on the transformation and upgradation of traditional industrial production lines.

Key words: Manufacturing Execution System; MES; Customized Production Line

目 录

第一章　引言

1.1　MES 系统的定义

1.1.1　先进制造研究机构给出的定义

制造执行系统（Manufacturing Execution System，MES）由美国的先进制造研究机构（Advanced Manufacturing Research，AMR）在 1990 年 11 月提出。AMR 将 MES 定义为位于上层的计划管理系统与底层的工业控制之间的面向车间层的管理信息系统。MES 为操作人员和管理人员提供订单的执行、跟踪以及包括人、设备、物料、客户需求等在内的所有资源的当前状态[1]。

1.1.2　制造执行系统协会给出的定义

制造执行系统协会（Manufacturing Execution System Association，MESA）将 MES 定义为在产品从订单发出到成品完工的过程中起到传递信息并优化生产活动的管理系统。在实时的生产过程中，MES 能够根据实时的情况，引导、发起、响应、报告生产活动，并能作出快速的响应以应对变化来减少无附加价值的生产活动，提高操作及生产流程的效率。MES 可以提升净利润、改善现金流和库存周转速度、保证按时出货。MES 还通过双向的直接通讯，在企业内部和整个产品供应链中提供有关产品行为的关键任务信息[2]。

1.2　MES 系统的功能及优势

1.2.1　MES 系统的功能

ANSI/ISA-95 是国际自动化学会（International Society of Automation，ISA）为开发企业和控制系统之间的自动化接口而制定的国际标准。该标准中提出并规定了如图 1-1 所示的普渡参考模型。此模型定义了从物理过程、智能设备、控制系统、制造信息系统到企业信息系统的五层（第 0 层到第 4 层）垂直系统架构[3]。

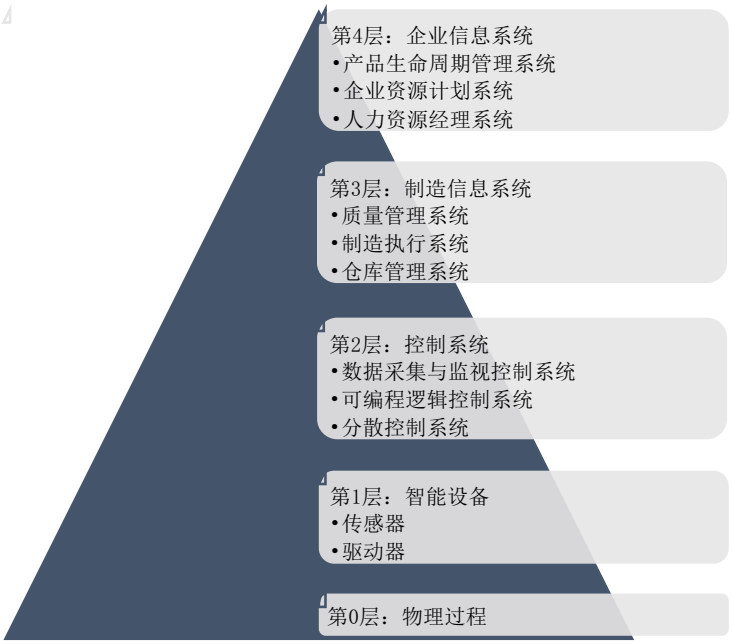


图 1-1　普渡参考模型

从该模型中可以看出，MES 系统位于第 3 级，被视为企业资源计划（Enterprise Resource Planning，ERP） 系统与数据采集与监视控制（Supervisory Control And Data Acquisition，SCADA）系统、可编程逻辑控制（Programmable Logic Controller，PLC）系统的连接中枢。其任务主要有以下 3 点：

① MES 系统要对整个车间制造过程进行优化；
② MES 系统要实时收集、分析和处理生产过程中数据；
③ MES 系统要与企业信息系统层（第 4 层）和控制系统层（第 2 层）进行信息交互，实现企业信息全集成。

MESA 在给出 MES 系统定义的同时，也给出了 MES 系统的 11 个功能，同时规定只要具备其中某一个或几个，也属于 MES 产品范畴。这 11 个功能分别是：

① 操作详细排序；
② 生产单元调度；
③ 产品追踪及溯源；
④ 人力资源管理；
⑤ 质量管理；
⑥ 维修管理；
⑦ 数据采集；
⑧ 过程管理；
⑨ 性能分析；
⑩ 文档控制；
⑪ 资源分配及状态管理[4]。

1.2.2 MES 系统的优势

MES 系统有助于创建完美无缺的生产过程，并有助于创建连续的生产数据视图。成功实施 MES 系统的其他优势包括：

① 生产可追溯；
② 确保准确的生产数据；
③ 减少停工时间，降低次品率，缩短配置时间；
④ 提高整体设备效能（Overall Equipment Effectiveness，OEE）；
⑤ 降低库存成本和压力，
⑥ 引进无纸化生产，
⑦ 对生产和其他方面进行准确经济评估[5]。

1.3 MES 系统的历史

自 1990 年 AMR 提出 MES 系统的概念以及 1997 年 MESA 提出 MES 系统的功能以来，MES 系统并没有想预期的那样发展的特别迅速，一开始也主要应用于汽车、半导体电子、烟草等自动化程度比较高的行业。2004 年，MESA 又提出了协同制造执行系统（C

ollaborative Manufacturing Execution Systems，C-MES）模型，强调了生产活动如何与业务运营相互作用，并研究了外包、供应链优化和资产优化等问题[5]。

随着不断的应用，MES 系统的行业性、专业性的需求与通用型的产品之间的矛盾也日渐突出，使得 MES 发展的十分缓慢。与 ERP 系统不同，MES 系统从产生开始，就是为了解决通用型的 ERP 功能难以解决的工厂管控问题，因此，即使有多年的发展，MES 系统也无法摆脱行业性、专业性对其形成的局限。

我国也在 20 世纪 90 年代开始了对 MES 系统以及 ERP 系统的跟踪研究、宣传和试点，而且提出了“管控一体化”“人、财、物、产、供、销”等颇具中国特色概念[1]。在 2007 年召开的中国共产党第十七次全国代表大会（十七大）上，胡锦涛总书记在十七大报告中指出要大力推进信息化与工业化融合[6]。在 2017 年召开的中国共产党第十九次全国代表大会（十九大）上，习近平总书记在十九大报告中也强调要加快建设制造强国，加快发展先进制造业，推动互联网、大数据、人工智能和实体经济深度融合[7]。

随着 2013 年德国提出的“工业 4.0”概念的不断发展，中国工业和信息化部（Ministry of Industry and Information Technology，MIIT）大力推进“工业化”与“信息化”融合，国务院也印发了《中国制造 2025》并作为国家行动纲领，旨在增强我国制造业综合实力。近些年来，随着准时制生产（Just in Time，JIT）模式、顺序生产（Just In Sequence，JIS）模式以及面向订单生产（Build to Order，BTO）模式等新型生产模式的提出，以及客户和市场对产品提出更高的要求，MES 系统被重新发现并得到重视。

有许多厂商推出了自己的 MES 系统产品以及其解决方案。这些厂商主要分为四类：第一类企业从自动化设备基础上发展而来，代表的有通用电气（General Electric，GE）、西门子（Siemens）、罗克韦尔（Rockwell）等；第二类企业是从 SCADA、HMI 发展而来，代表的有阿达斯特拉科技（AdAstrA）、剑维软件（AVEVA）、CI Tech 等；第三类企业是一开始就专注于特定行业的 MES 系统或者 MES 系统中的某一项功能，代表的有艾斯本（AspenTech）、PSI、宝信软件（Baosight）等；第四类企业是由 ERP 系统厂商向下发展来的，代表的有思爱普（SAP）、用友（yonyou）等[8]。

1.4 MES 系统的应用概述

1.4.1 MES 系统在某电子信息装备企业的应用

J 公司是一家大批量批次生产模式的电装行业企业，生产的产品主要是 LED 表贴产品。J 公司根据实际情况，设计并应用了包含批次任务控制模型的 MES 系统。其业务系统通过“模型-视图-控制器”架构（Model-View-Controller，MVC）设计，采取前后端分离模式，前端采用 JAVA 服务器页面技术（Java Server Pages，JSP）完成数据请求和页面渲染，后端采用 Spring 组件完成请求响应和逻辑处理，实现 MES 系统中的各项业务。该 MES 系统包含如图 1-2 所示的 7 个模块及其功能。

J 公司导入 MES 系统解决方案后提升了计划效率和正确性，优化了生产现场组织管理，提高了车间生产安排有序性、合理性，提升了产能，提高了企业整体生产效率和交付

准确率[9]。

图 1-2　J 公司的 MES 系统总体架构和模块图

1.4.2　MES 系统在某半导体封装企业的应用

某半导体封装企业的 MES 系统架构实现主要利用 C/S 三层框架模式（包含数据访问层、表示层和业务逻辑层）来实现数据库、客户端、服务器端的开发。客户端主要负责接收业务逻辑层的数据处理操作、用户请求的数据交由业务逻辑层处理并将请求结果返回显示在客户端图形界面。由于系统客户端的图形界面众多，因此使用 Visual Studio 集成的控件库、第三方控件库 Spread、Active Thread。服务端主要负责业务逻辑层、数据访问层的实现，业务逻辑层需调用 Server 函数库实现，数据访问层需调用 DBCommon 函数库实现。DBCommon 函数库主要负责访问 Oracle 数据库进行增删改查等一系列数据操作。而 Server 函数库主要负责判端 DBCommon 函数库是否将访问数据库的结果返回给客户端。客户端与服务器端之间使用 Highway 101 进行通信，而数据库选用的是 Oracle[10]。

1.5　MES 系统的设计概述

本课题的主要研究内容是定制化加工生产线的 MES 系统设计。根据前人的研究和实际控制需求，本课题的 MES 系统将采用前后端分离的 C/S 架构，前端包括用于交互的 Web 页面和现场监控大屏程序，后端为用于控制现场生产线设备和响应前端请求的服务器（后文称之为 MES 服务器），除此之外，还有负责存储数据的数据库。课题主要研究内容如图 1-3 所示。

本课题将主要运用 MES 技术、 PLC 控制技术、工业网络通信技术等实现对基于 S7-1500 的定制化加工生产线的工作流程和信息交互进行控制，基本思路如下：

① 分析定制化加工生产线模块化组合、标准化架构，进行控制逻辑关系分析，提出 MES 系统的控制需求；

② 收集 MES 系统设计与开发的相关资料，比较各个 MES 系统解决方案的优缺点，提出整体方案；

③ 设计 MES 系统的整体架构，完成智能物料分拣系统的生产流程图、MES 软件部署图、用户界面设计图等设计；

④ 编写 MES 系统的程序，实现 MES 系统各个模块的功能，完成 MES 系统各个模块的调试；

⑤ 将程序发布到服务器和移动终端，完成定制化加工生产线在 MES 系统控制下的现场调试和试运行。

图 1-3　MES 系统框架

另外，为了适应企业的具体使用环境，本课题的所有程序均需支持跨平台运行，主要包括 X86-64（又称 X64 或 AMD64）架构和 ARM64 架构的 Windows、Linux、macOS。

本课题的设计流程如图 1-4 所示。

图 1-4　课题的设计流程

本课题所设计的 MES 系统的优势如下：

① 与生产线上的 PLC 程序紧密配合，提供用户管理、订单排序、生产控制、设备监控、物料管理等 MES 系统的主要功能；

② 采用跨平台程序开发框架，课题中所有程序均支持主流操作系统及架构，满足企业的各种需求；

③ 采用前后端分离设计，后端 MES 服务器运行时无图形页面，降低了性能需求和能耗，前端包括 Web 页面与现场监控大屏程序，其中 Web 页面无需安装，在浏览器中访问即可，现场监控大屏程序仅在生产线所在车间部署，用于实时监控；

④ 采用模块化设计，降低了程序的复杂度，提高了代码的充用率，使得程序易于维护、修改和功能扩充，当生产线的动作单元变动或者用于控制其他类似生产线时仅需修改少量代码即可满足控制要求；

⑤ 使用多种数据加密算法，保证了数据安全；

⑥ 提供跨平台的编译脚本，在任何操作系统上都可以对本课题中的程序进行编译并打包。

第二章　课题研究分析

2.1　被控对象分析

本课题所控制的定制化加工生产线如图 2-1 所示，其包括了供料单元、罐装单元、压合单元、分类输出单元等 4 个独立的工作单元，单元之间没有信息交互，并且每个单元都有默认模式（Default Mode）和 MES 模式（MES Mode）2 种运行模式。每个单元通过传送带进行连接，生产线投入运行后，传送带带动托盘周而复始的运动。

各单元均采用西门子 S7-1516 PLC 作为主控制器，使用西门子 XB008 工业网络交换机与变频器、伺服驱动器、分布式 I/O、HMI 以及上位机进行 PROFINET 以及 TCP/IP 通讯，使用西门子 G120 变频器驱动传送带的三相异步电机，使用西门子 V90 伺服驱动器控制抓取托盘用的伺服电机，使用西门子 ET200SP 分布式 I/O 来采集各传感器的数据。ET 200SP 上还安装了 IO-LINK 模块，用于连接 RFID 读写器和按钮。此外，各单元还配有西门子 HMI 以供调试和在默认模式下下达订单。供料单元如图 2-2 所示，其他单元与之类似。

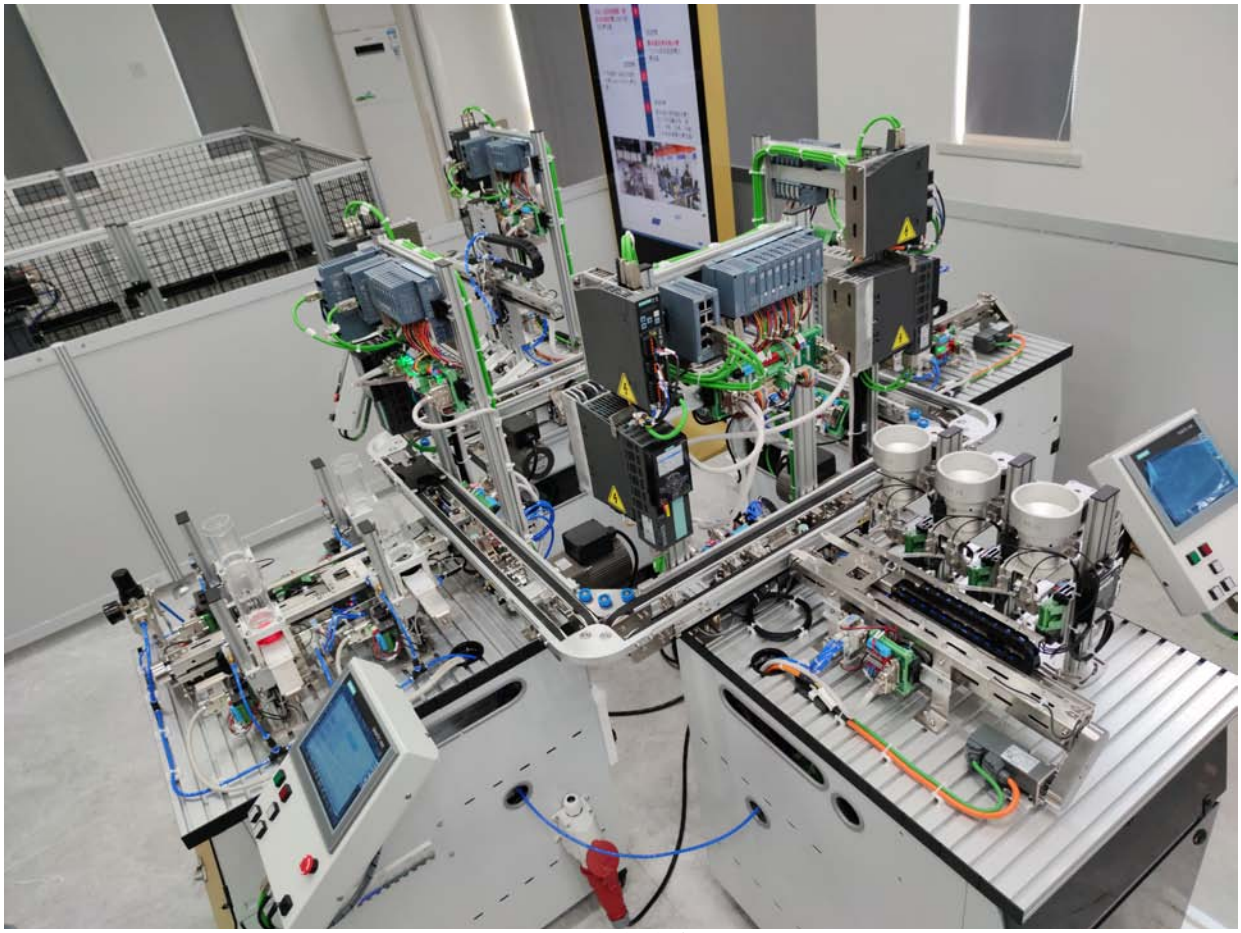


图 2-1　生产线实物图

如何对生产线上的 PLC 进行控制是本课题重点研究的方向。通过组内讨论，最终确定了如下所述的技术路线。

在默认模式下，各单元通过读取生产线上的托盘内置的 RFID 电子标签中的数据来决

定本单元是否工作以及确定工作的参数，此时 MES 系统仅可监测设备的运行状态。通过 TCP/IP 连接到上位机后，PLC 将会每隔 1 秒向上位机的 2001 端口发送一次设备状态数据。设备状态数据为 8 位 16 进制整数，如表 2-1 所示。

例如，PLC 向 MES 的 2001 端口发送了 0x00050281（0x81 转换为二进制则为 0b10000001），其含义为该PLC 的设备编号为 5，类型为 Siemens，当前处于 MES 模式下并自动运行。

在 MES 模式下，PLC 除了会每隔 1 秒向上位机的 2001 端口发送一次设备状态数据外，当托盘到达该单元时还会向上位机的 2000 端口发送一次 256 位 16 进制的设备控制数据，MES 系统接收后要回复 180 位以上的 16 进制的设备控制数据，PLC 根据收到的设备控制数据来决定本单元是否工作以及确定工作的参数。设备控制数据为如表 2-3 所示。

MES 向 PLC 发送设备控制数据时，若无需操作参数，则发送前 180 位数据，若需要操作参数，则在添加 76 个 0 后添加操作参数，操作参数均为 8 位 16 进制整数。PLC 向 MES 发送设备控制数据时需发送前 256 位。

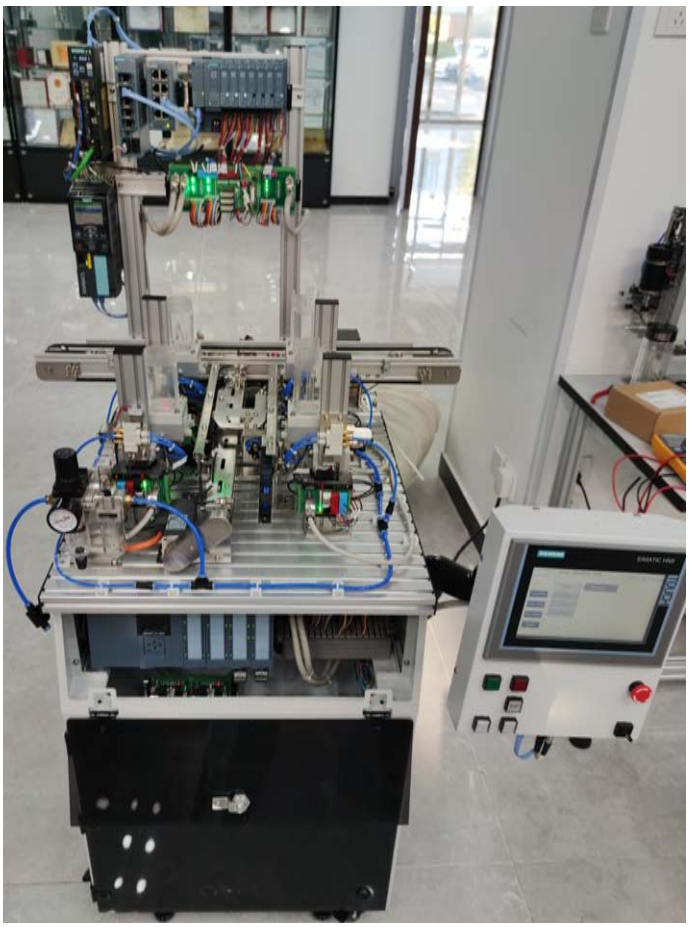


图 2-2　供料单元

表 2-1　设备状态数据说明

位数	说明	值
0-3	设备编号	
4-5	PLC 类型	0x01 为 CODESYS，0x02 为 Siemens
6-7	设备状态	转换为 8 位 2 进制数，见表 2-2

表 2-2　设备状态数据中的 6-7 位说明

位数	对应 2 进制位数	说明	值
6	0	MES 模式	1 为是 0 为否
	1	错误 0	
	2	错误 1	
	3	错误 2	
7	4	复位	
	5	繁忙	
	6	手动	
	7	自动	

表 2-3　设备控制数据说明

位数	标识	说明	值
0-7	TcpIdent	校验信息	MES 向 PLC 发送时为 0x33333333 PLC 向 MES 发送时为 0x33333302
8-11	RequestID	设备编号	
12-15	mClass	功能编号 1	值及其对应功能见附录
16-19	mNo	功能编号 2	
20-23			4 个 0
24-27	DataLength	操作参数数据长度	操作参数的字节数
28-31	ResourceID	设备编号	
32-39	ONo	订单编号	
40-43	OPos	子订单编号	
44-47	WPNo	计划编号	0x0001
48-51	OpNo	操作编号	0x0001
52-71			20 个 0
72-79	PNo	成品编号	
88-91	StepNo	步骤编号	0x0001
92-175			84 个 0
176-177	Data1		0x15
178-179	Data2		0x16
180-255			76 个 0
256-263	refData1	操作参数 1	
264-271	refData2	操作参数 2	
……	……	……	
	refDataN	操作参数 N	

2.2　模块化程序设计

模块化程序设计是指在进行程序设计时将一个大程序按照功能划分为若干小程序模块，每个小程序模块完成一个确定的功能，并在这些模块之间建立必要的联系，通过模块的互相协作完成整个功能的程序设计方法。模块化的目的是为了降低程序复杂度，使程序设计、调试和维护等操作简单化。

模块化程序设计的基本思想是自顶向下、逐步分解、分而治之，即将一个较大的程序按照功能分割成一些小模块，各模块相对独立、功能单一、结构清晰、接口简单。模块化程序设计应该遵循以下几个主要原则：

①模块独立，模块与其他模块的联系应该尽可能得简单，各个模块具有相对的独立性；

②模块的规模要适当，如果模块的功能太强，可读性就会较差，若模块的功能太弱，就会有很多的接口。

③分解模块时要注意层次。

本课题中的 MES 服务器、Web 页面、现场监控大屏程序均采用模块化程序设计来降低程序复杂度和提高代码充用率，使得程序易于维护、修改和功能扩充。由于网络通讯、数据加密、数据管理（不含数据库访问）等3 个功能在这本课题中使用的频率非常高，并且这 3 个功能之间互相独立，功能比较单一，所以将这三个功能分别做共用模块，即网络通讯模块、数据加密模块、数据管理模块（不含数据库访问模块），以便在不同程序之间使用相同的功能和接口。在开发 MES 服务器、Web 页面、现场监控大屏程序时除了使用上述的 3 个公共模块外，每个程序还将拥有自己特有的模块，如 MES 服务器中的数据库访问模块、生产控制模块，Web 页面的后台服务模块等。

2.3 数据库的选型

数据库（Database）是建立在计算机存储设备上的，按照数据结构来组织、存储和管理数据的仓库，目前主要分为关系型（Relational）数据库和非关系型（Non-relational）数据库。由于关系型数据库使用结构化查询语言（Structured Query Language， SQL），所以又被称为 SQL 数据库；而非关系型数据库又被称为 NoSQL（Not only SQL）数据库。其中非关系型数据库又分为键值存储（Key-Value Stores）数据库、列式存储（Wide Column Stores）数据库、文档型（Document Stores）数据库、图形（Graph）数据库等。常见的数据库及其排名如表 2-5 和图 2-3 所示[11]。

虽然非关系型数据库有易扩展、高性能等优势，但是从图表中可以看出，目前主流应用的主要还是关系型数据库，这得益于SQL 语言的通用性，操作更加方便。还可以看出，Oracle、Mysql 与 SQL Server（均为关系型数据库）的流行度远超其他数据库。

表 2-5 常见数据库及排名

Rank			DBMS	Database Model	Score		
Apr 2021	Mar 2021	Apr 2020			Apr 2021	Mar 2021	Apr 2020
1.	1.	1.	Oracle	Relational, Multi-model	1274.92	-46.82	-70.51
2.	2.	2.	MySQL	Relational, Multi-model	1220.69	-34.14	-47.66
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1007.97	-7.33	-75.46
4.	4.	4.	PostgreSQL	Relational, Multi-model	553.52	+4.23	+43.66
5.	5.	5.	MongoDB	Document, Multi-model	469.97	+7.58	+31.54
6.	6.	6.	IBM Db2	Relational, Multi-model	157.78	+1.77	-7.85
7.	7.	↑ 8.	Redis	Key-value, Multi-model	155.89	+1.74	+11.08

Oracle 数据库（Oracle Database，Oracle）是由甲骨文公司（Oracle）推出的一款关系型数据库（如无特殊说明，后文中出现的 Oracle 均指 Oracle 数据库而非甲骨文公司）。Oracle 在数据库领域一直处于领先地位，是目前最流行的数据库管理系统。其支持的操作系统有 Microsoft Windows x64、Linux x86-64、Oracle Solaris 等，使用它的企业有花旗（Citi）、松下（Panasonic）等。Oracle 获得来最高认证级别的 ISO 标准认证，并且性能也是所有关系型数据库中最高的，同时它也对硬件要求很高，安装包体积略大（Oracle 19c

在 Windows 上的安装包体积为 2.9GB，在 Linux 上为 2.8GB），且不支持 macOS，不符合本课题的设计要求。

SQL Server 是微软公司（Microsoft）推出的一款关系型数据库，并且过去九年来被美国国家标准与技术研究院（National Institute of Standards and Technology，NIST）评为漏洞最少的数据库。其支持的操作系统有 Windows 和 Linux，也支持在 Docker（一个开源的应用容器引擎）中部署，并且在 Windows 平台上提供了诸多版本，除了收费的 Enterprise 版、Standard 版和 Web 版外，还有免费使用的 Developer 版、Express 版和超轻量化的 Express LocalDB 版，但是其不支持 ARM 架构，不符合本课题的设计要求。

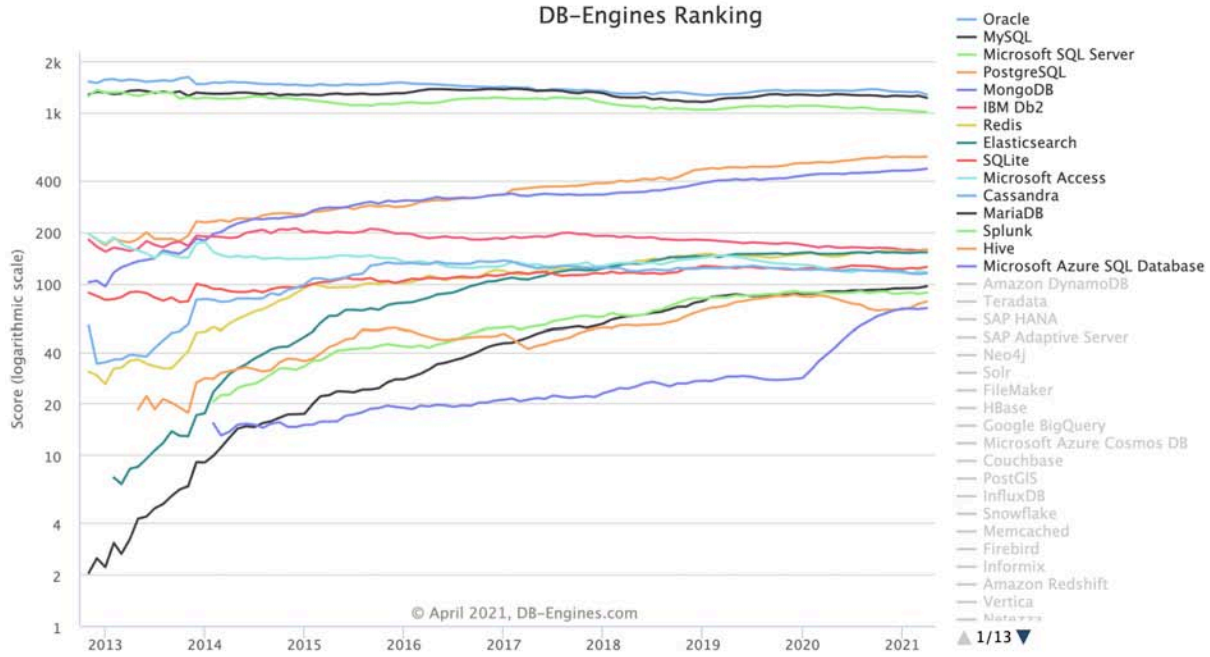


图 2-3　常见数据库及其排名

MySQL 是 MySQL AB 公司（已被甲骨文公司收购）开发的一款关系型数据库，其速度是所有数据库里最快的，被广泛应用于 Web 应用。由于其开放源代码，因此可以在任何架构和操作系统上编译并使用。本课题的数据库使用 MySQL Community Server 8 进行开发，数据库中将包含用户账户、现场设备、原材料以及客户订单等数据的管理。

2.4　MES 服务器与 WEB 页面所用框架的选择

.NET（dotnet）是一种用于构建多种应用的免费开源开发平台，可开发的程序有：控制台应用、桌面应用、Web 应用、移动应用、嵌入式应用、机器学习等。它的特点是跨平台、开源、高效、生态丰富，可以为许多操作系统以及众多处理器架构创建应用。其支持的操作系统有：Windows、macOS、Linux、Android、iOS、tvOS、watchOS 等，其支持的处理器体系结构有：X64、X86、ARM32、ARM64 等。

除了使用如操作系统 API 特定于平台的功能，其余的代码在任何系统上几乎是一样的，并且使用类库在不同应用和应用类型中共享功能。.NET 提供了 3 种编程语言，分别是 C#、F#和 Visual Basic。C#作为.NET 平台的一种新式编程语言，不仅面向对象，还类

型安全。C#源于 C 语言系列，C、C++、Java 和 JavaScript 程序员很快就可以上手使用。多项 C# 功能有助于创建可靠且持久的应用程序：

① 垃圾回收（Garbage Collection，GC）：自动回收不可访问的未用对象所占用的内存。

② 异常处理（Exception Handling）：提供了一种结构化且可扩展的方法来进行错误检测和恢复。

③ Lambda 表达式：提供函数编程技术的支持。

④ 语言集成查询 (Language-Integrated Query，LINQ)：用于处理来自任何源的数据。

⑤ 异步操作：提供用于构建分布式系统的语法。

所有 C#类型（包括 int 和 double 等基元类型）均继承自 object 根类型，所有类型共用一组通用运算。任何类型的值都可以一致地进行存储、传输和处理。

ASP.NET Core 是一个跨平台的开源框架，属于.NET 平台，用于在 Windows、macOS 或 Linux 上开发基于云的新式 Web 应用，并且支持Blazor。Blazor 是一个使用.NET 生成交互式客户端 Web UI 的框架，使用 ASP.NET Core 和 Blazor 进行 Web 开发可提供以下优势：

① 可使用 C#代替 JavaScript 来编写代码并创建信息丰富的交互式 UI；

② 可利用现有的.NET 库生态系统；

③ 可在服务器和客户端之间共享应用逻辑；

④ 可使用 Visual Studio 中的 IntelliSense 智能提示和纠错功能在开发中保持高效工作。

OpenJDK 是 Java 平台的开源的软件开发工具包（Development Kit），是整个 Java 开发的核心，包含了 Java 的运行环境和开发工具。Java 既是一个平台，也是一门编程语言。Java 平台提供了 2 个组件：Java 虚拟机（Java Virtual Machine，JVM）和 Java 应用程序编程接口（Java Application Programming Interface，Java API）。JVM 是 Java 平台的基础，并移植到各种系统上，如 Windows、macOS、Linux 等，所以使用 Java 开发的程序亦能很好的支持跨平台。

Java 是一门面向对象编程语言，不仅吸收了 C++语言的各种优点，还摒弃了 C++里难以理解的多继承、指针等概念，因此 Java 语言具有功能强大和简单易用两个特征。Java 可以编写桌面应用程序、Web 应用程序、分布式系统和嵌入式系统应用程序等。Java 具有简单性、面向对象、分布式、健壮性、安全性、平台独立与可移植性、多线程、动态性等特点。

Java Server Pages（JSP）是一种动态网页技术标准。JSP 技术是以 Java 语言作为脚本语言的，JSP 网页为整个服务器端的 Java 库单元提供了一个接口来服务于 HTTP 的应用程序。JSP 开发的 WEB 应用可以跨平台使用。JSP 将 Java 代码和特定变动内容嵌入到静态的页面中，实现以静态页面为模板，动态生成其中的部分内容。

作为一个独立于硬件平台的环境，Java 平台在运行时比本地代码慢一点。并且根据 TechEmpower 提供的 Web 框架性能基准测试的数据来看，使用 Java 开发的网页平均每秒能够响应 217 万次请求，而使用.NET 开发的网页平均每秒能够响应 701 万次请求。

本课题的 MES 服务器和 Web 页面将基于.NET 6.0 平台，采用 C#语言进行开发，并且共用网络通讯模块、数据加密模块、等基础模块。本课题的 MES 服务器使用 MySQL Connector/NET（NuGet 包名为 MySql.Data）来连接数据库进行数据的读写操作，通过 TCP/IP 协议进行 MES 服务器与 WEB 页面、MES 服务器与现场 PLC 等的通讯。本课题的 Web 页面还将采用 ASP.NET Core 6.0 框架和 Blazor 框架，并使用 Razor 标记开发网页界面，同时还将使用 Bootstrap Blazor 组件库（NuGet 包名为 BootstrapBlazor）开发部分页面组件。MES 服务器和 WEB 页面相互配合，实现的功能有用户管理、订单排序、生产控制、设备监控、物料管理等。MES 服务器与 Web 页面之间的通讯将会采用 Base64 编码和随机密钥的 AES 加密来保证数据安全。

2.5 现场监控大屏程序所用框架的选择

Electron 是一个基于 Chromium 和 Node.js，能使用 JavaScript、HTML 和 CSS 等 Web 技术来创建原生桌面应用程序的开源框架。这些应用程序被打包后可以直接在 Windows、macOS、Linux 上运行。使用 Electron 开发的软件有 Microsoft Teams、Visual Studio Code（VS Code）、Facebook Messenger 等。

Electron.NET 是在 Electron 应用程序中嵌入 ASP.NET Core 应用程序的适配器。通过 Electron.NET IPC，可以从.NET 中调用 Electron API。

本课题的现场监控大屏程序将基于.NET 6.0 平台、ASP.NET Core 6.0 框架、Blazor 框架和 Electron.NET 框架，采用 C#语言进行开发，采用 Razor 标记构建用户界面，使用与 MES 系统和 Web 页面相同的网络通讯模块、数据加密模块等基础模块。现场监控大屏程序将主要用来监控生产线设备的运行状态。

2.6 加密算法及其原理

2.6.1 AES 加密

高级加密标准（Advanced Encryption Standard，AES）是一种区块加密标准，属于对称加密。AES 采用 Rijndael 加密法，由比利时密码学家 Joan Daemen 和 Vincent Rijmen 设计。AES 用来替代原先的数据加密标准（Data Encryption Standard，DES）。AES 的区块长度固定为 128 位（16 字节），密钥（Key）长度可以是 128 位（16 字节），192 位（24 字节）或 256 位（32 字节），支持的加密模式有密码块链（Cipher Block Chaining， CBC）模式、密码反馈（Cipher Feedback，CFB）模式、密码文本窃用（Cipher Text Stealing，CTS）模式、电子密码本（Electronic Codebook，ECB）模式和输出反馈（Output Feedback，OFB）模式。Key 的长度不同，推荐加密轮数也不同，如 128 位 Key 推荐进行 10 轮加密，256 位 Key 推荐进行 14 轮加密。

在进行每一轮的 AES 加密时，首先将明文划分成明文块，明文块用 p 表示，每块 16

字节，不足 16 字节的在后面补 0。例如明文为“abcdefghijklmnopqrstuvwxyz”26 个英语字母，在字符编码中一个英语字母占 1 字节，则其被分成“abcdefghijklmnop”与“qrstuvwxyz”两个明文块。

CBC 模式引入了反馈，每个明文块在加密前，通过异或操作与前一个密文块。这样确保了即使明文包含许多相同的明文块，这些相同的明文块也会加密为不同的密文块。在加密第一个明文块的时候使用初始化向量（initialization vector，IV）通过异或操作与第一个明文块结合，如图 2-4 所示，公式见式 2-1。

$$\begin{cases} c_1 = E_{CBC}[(p_1 \oplus IV), Key] \\ c_n = E_{CBC}[(p_n \oplus c_{n-1}), Key] \quad (n = 2,3,4,\cdots) \end{cases} \tag{式 2-1}$$

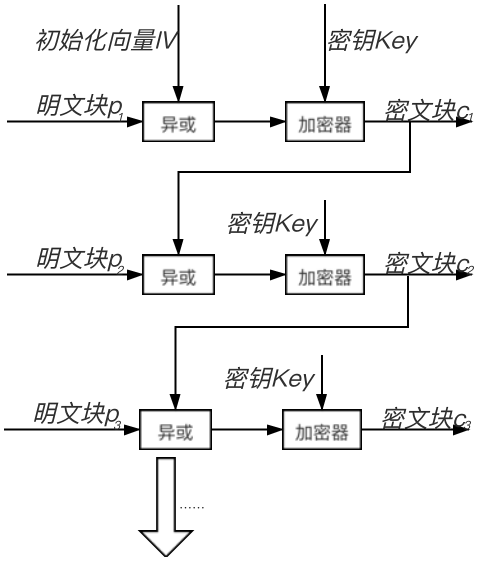


图 2-4　CBC 模式加密过程

把密文块依次组合起来即得到该轮加密的密文 C_n，每一轮加密后得到的密文为下一轮加密的明文，最后一轮加密后得到的密文即为整个 AES 加密后的密文 C。AES 加密器将输入矩阵依次进行密钥加、字节代换、行移位、列混合（最后一轮时替换成密钥加）、的操作后得到输出矩阵。由于过程繁杂，不在过多阐述。

AES 解密时执行与加密时相反的操作和相同的轮数。在进行每一轮的 AES 解密时，首先将密文划分成若干个密文块，每块 16 字节。对于 CBC 模式，每轮的解密过程如图 2-5 所示，公式见式 2-2。

$$\begin{cases} p_1 = D_{CBC}(c_1, Key) \oplus IV \\ p_n = D_{CBC}(c_n, Key) \oplus c_{n-1} \quad (n = 2,3,4,\cdots) \end{cases} \tag{式 2-2}$$

把明文块依次组合起来即得到该轮解密的明文 P_n，每一轮解密后得到的明文为下一轮解密的密文，最后一轮解密得到的明文即为整个 AES 解密后的明文 P。AES 解密器将输入矩阵依次进行逆列混合（第一轮时替换成密钥加）、逆 行移位、逆字节代换、密钥加的操作后得到输出矩阵。

本课题中的 AES 加密采用最安全的 256 位密钥和 CBC 模式。当有客户端（包括 Web 网页和现场监控大屏程序）请求连接时，MES 服务器会随机生成 256 位的 128 位的 IV，

并通过网络发送给客户端。之后每次收发信息均采用此 Key 和 IV 进行 AES 加解密，直到客户端关闭。为了保证安全，在发送 Key 和 IV 时，还将对 Key 和 IV 进行 RSA 加密，并由客户端解密。MES 服务器与客户端收发加密的信息还需进行 Base64 编解码，如图 2-6 所示。RSA 加密和 Base64 编码将在下文中介绍。

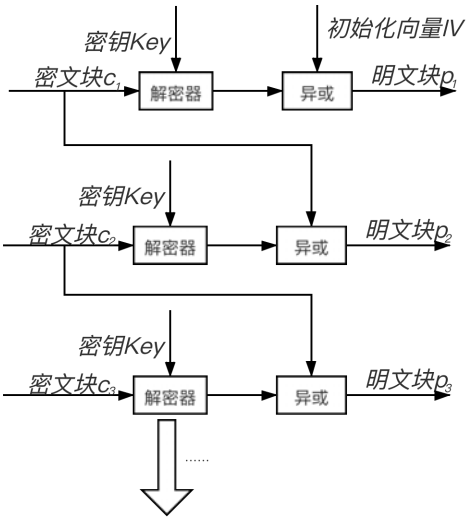


图 2-5　CBC 模式解密过程

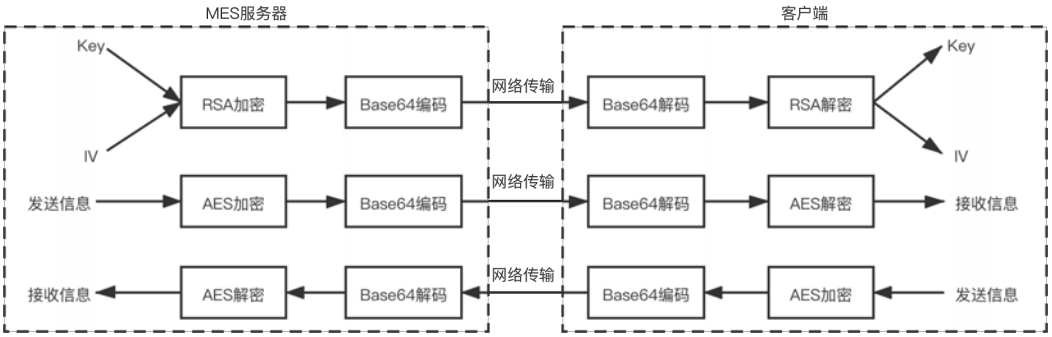


图 2-6　通讯信息加解密过程

2.6.2　RSA 加密

RSA 加密是一种非对称加密的方法，于 1977 年由麻省理工学院（Massachusetts Institute of Technology，MIT）的罗纳德·李维斯特（Ronald Rivest）、阿迪·萨莫尔（Adi Shamir）和伦纳德·阿德曼（Leonard Adleman）一同提出。RSA 加密算法的密钥分为公钥（Public Key，pubKey）和私钥（Private Key，priKey），pubKey 的长度可选 512 位、1024 位、2048 位、4096 位。在加密时可以选择用公钥加密、用私钥解密，也可以选择用私钥加密、用公钥解密。不论选取哪种方法，如果密文和加密密钥都暴露了的话，也是无法解密得到明文的。

RSA 算法的大致过程如下：

①随机选择两个不相等的质数 p 和 q，越大越好。

②计算 p 和 q 的乘积 n，见式 2-3。

$$n = p \times q \qquad \text{（式 2-3）}$$

③计算 n 的欧拉函数值 ϕ(n)，见式 2-4。

$$\varphi(n) = (p-1)(q-1) \quad \text{（式 2-4）}$$

n-1 为明文的最大值，若明文过大需要对其进行分割并分别加密。

④随机选择一个整数 e，并使 e 满足式 2-5。

$$\begin{cases} 1 < e < \varphi(n) \\ e\text{与}\varphi(n)\text{互质} \end{cases} \quad \text{（式 2-5）}$$

⑤计算 e 对于 ϕ(n)的模反元素 d，见式 2-6。

$$e \times d \,\%\, \varphi(n) = 1 \quad (\%\text{为模运算，下同}) \quad \text{（式 2-6）}$$

即

$$d = \frac{1 + i \times \varphi(n)}{e} \quad (i = 1,2,3,\cdots) \quad \text{（式 2-7）}$$

⑥使用公钥加密并用私钥解密，见式 2-8。

$$\begin{cases} C = P^e \,\%\, n \\ P = C^d \,\%\, n \end{cases} \quad (P\text{为明文，}C\text{为密文}) \quad \text{（式 2-8）}$$

例如，选取 p=3，q=11，则 n=33=0b100001，ϕ(n)=20，e 的取值范围为{3,7,9,11,13,17,19}。选取 e=9，d=29。设有一明文为“abc”，则 P=0x616263，每位十六进制数进行一次划分并使用公钥进行加密，则密文C=0x18011811180F，解密后的明文P'=0x616263。其中，0x6 的 9 次方除以 33 取余的结果为 0x18，0x18 的 29 次方除以 33 取余的结果为 0x6，其余类似。

上述中的 e 为公钥，d 为私钥。在实际应用中，n 与 e 被写进公钥文件中，n 与 d 被写入私钥文件中。使用相应的密钥文件即可进行加解密操作。本课题使用在线工具创建 2048 位的公钥及其对应的私钥。由于 RSA 算法进行的都是大数计算，无论是软件还是硬件实现都要比 AES 慢，所以本课题只用 RSA 算法进行少量数据加密，如对 AES 算法的 Key 和 IV 加密。

2.6.3 Base64 编码

Base64 编码是一种基于 64 个可打印字符（包括数字、大小写字母和符号“+”“/”）来表示二进制数据的方法。Base64 要求把每 24 位（3 字节）的数据分成4 组，每组 6 位，然后每组再在最高位添加 2 个 0，组成 32 位（4 字节）。例如，有一明文为“abc”，用二进制表示则为 0b01100001_01100010_01100011，每 6 位分成一组，则为 0b011000_010110_001001_100011，每组的最高位再添加 2 个 0，则为 0b00011000_00010110_00001001_00100011，按照 Base64 索引表将结果转位字符串，则密文为“YWJj”。

2.7 数据转换

本课题中关键的技术点在于使同一份数据在数据库中、网络传输中以及程序运行中保持一致。在 C#程序运行中，数据以类的实例的形式存在于内存中。在网络传输过程中，数据以字符串，甚至是字节数组的形式存在于网络中。在数据库中，数据为一条记录。

本课题中的数据主要有用户、设备（主要指 PLC）、物料、产品、订单等。在数据管

理模块中，以上类别的数据都会被设计成若干个类（class），每个类的属性基本都与数据库中的字段相对应。由于 C#的类默认含有 ToString()方法，所以每个类都需要重写 ToString()方法以便将该实例转化为特定字符串，将字符串逆转化为类的实例的方法为静态方法 FromString()，在每个类中均有定义，详见下文。有了 ToString()和 FromStirng()这两个方法，即可方便的在网络中传送实例。例如 MES 服务器如果向 Web 页面发送某个实例时，调用该实例的 ToString()方法得到字符串，经加密后发送给 Web 页面即可；Web 页面收到后首先解密，然后调用类的 FromString()方法并将得到的字符串作为参数传给该方法即可得到与 MES 服务器中相同的实例。整个过程如图 2-7 所示。

在数据库中，上述类别的数据会被设计成若干个数据表，每个数据表中的字段基本都与数据管理模块中类的属性相对应。但在数据管理模块的产品类中，需要指定所需的物料及其数量，并且至少需要 1 种物料。为了便于管理，在产品类中的物料属性的数据类型为列表，列表中的元素为物料类的实例。但在数据库中不存在如列表之类的数据类型，所以在产品数据表中将该字段的数据类型设置为文本。将产品实例存放进数据库时，将遍历物料列表中的物料实例，并调用物料实例的 ToString()方法将该实例转化成字符串，在字符串的开头添加上该字符串的长度，将得到的字符串连接起来并用分号隔开，即可得到用来描述物料列表的字符串，最后，将该字符串以及其他属性存放进数据库即可。在从数据库读取产品实例时，从物料字段获取用于描述该产品的物料列表的字符串，通过循环解析出全部的字符串片段，然后依次调用物料类的 FromString()方法将该字符串片段作为参数传入即可得到物料类的实例，最后依次将得到的物料实例添加进该产品实例的物料列表即可。整个过程如图 2-8 所示，具体实现详见下文。另外，订单类中也有类似的设计。

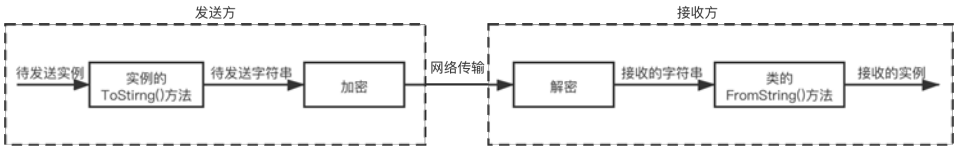


图 2-7　网络收发实例

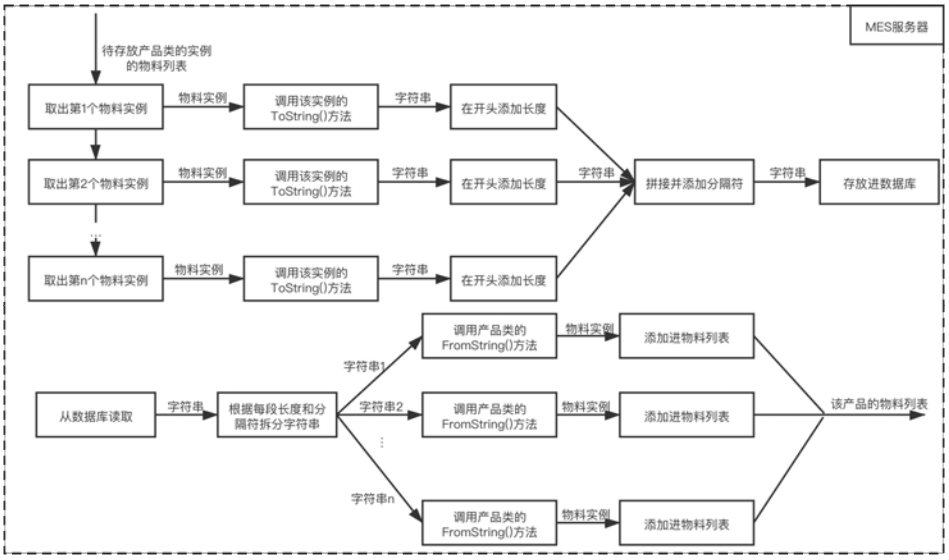


图 2-8　产品类的存放与读取

在处理 PLC 发送来的设备状态数据时也需要对其进行处理。发送来的数据为 8 位的 16 进制整数，在 C#程序中会被识别成含有 4 个元素的字节数组（C#类型为 byte[]，长度为 4）。数组的前两个元素为设备编号，将第0 个元素乘以 256 之后再加上第 1 个元素即可得到设备编号。数组的第 2 个元素为 PLC 类型，如果该字节的值为 1 则 PLC 的类型为 CODESYS，如果该字节的值为 2 则 PLC 的类型为 Siemens。

数组的第 3 个元素即为设备当前的状态，由于数据在内存中本身就以二进制存储，所需不再需要手动将数据由 16 进制转换成 2 进制。1 字节对应 8 位二进制数，1 个布尔类型的数对应 1 位二进制数，将该字节的每一位都解析成布尔类型（C#类型为 bool）即可得到上表 2-2 中的 8 种状态，见式 2-9。

$$b = B \div 2^{7-i} \,\%\, 2 \quad (7 \leq i \leq 0) \qquad \text{（式 2-9）}$$

其中，B 为待解析的字节，i 为要解析的位数（从左往右数并且从 0 开始），b 为解析的结果。

在 C#中，若除法运算符“/”两侧的数中有一个不为小数，则运算结果也不为小数。上式中，字节数据B 会被处理成整数，而 2 的非负整数次幂也不为小数，则商也为整数，并且舍弃小数部分（即不会对小数部分进行四舍五入）；再将商除以 2 并取余数即可得到第 i 位的值，例如，设 B=0x8A=0b10001010，若需要取其第 4 位的数，则先将 B 除以 2 的立方，即 $0b10001010/2^3$，得到商 0b10001，再将 0b10001 除以 2，商 0b1000 余 1，即结果 b 为 1（true）。

第三章　数据库设计

3.1　数据库总体设计

打开终端并登录 MySQL 监视器，执行 SQL 语句“*CREATE DATABASE slone_mes_db;*”创建本课题所需的名为“slone_mes_db”的数据库。根据课题需要，在 slone_mes_db 数据库中建立如表 3-1 所示的数据表。

表 3-1　数据库中各表的设计

表名	描述
user	用户管理
inventory	物料管理
plc	设备管理
product	产品管理
order	订单管理

下文将介绍所有数据表的设计与实现。

3.2　用户管理数据表的设计

在本课题中，一个用户拥有如表 3-2 所示的属性。

表 3-2　用户管理数据表的设计

属性名	数据类型	完整性约束条件
username	varchar(8)	PK, NN, UQ
password	varchar(16)	NN
role	varchar(8)	NN
description	varchar(128)	

其中，属性 username 表示一个用户的名称，不超过 8 个英语字符，按照控制需求，需要把用户名设置为主键（Primary Key， PK），即可以通过用户名来查找某一用户，用户名不能为空（Not Null， NN）并且不能重复（unique，UQ）。

属性 password 为该用户的登录密码，最多 16 个英语字符，并且设置为 NN。

属性 role 为该用户的权限，最多为 8 个英语字符，并且设置为 NN。本课题的 MES 系统将为不同权限的用户提供不相同的功能，可选的值有 admin、other 等。

属性 description 为该用户的备注信息，最多 128 个英语字符和 64 个汉字。

根据上述设计，确定要执行的 SQL 语句为：“

```
CREATE TABLE `user` (
  `username` VARCHAR(8) NOT NULL,
  `password` VARCHAR(16) NOT NULL,
  `role` VARCHAR(8) NOT NULL,
  `description` VARCHAR(128) NULL,
  PRIMARY KEY (`username`),
  UNIQUE INDEX `username_UNIQUE` (`username` ASC) VISIBLE);”
```

执行结果如图 3-1 所示。

向用户表中添加一个管理员用户，用户名为 admin，密码为 123456789，权限为 adm

in，描述为 Administrator，要执行的 SQL 语句为：“

INSERT INTO `user` (`username`, `password`, `role`, `description`)

VALUES ('admin', '123456789', 'admin', 'Administrator');”

执行的结果如图 3-2 所示。

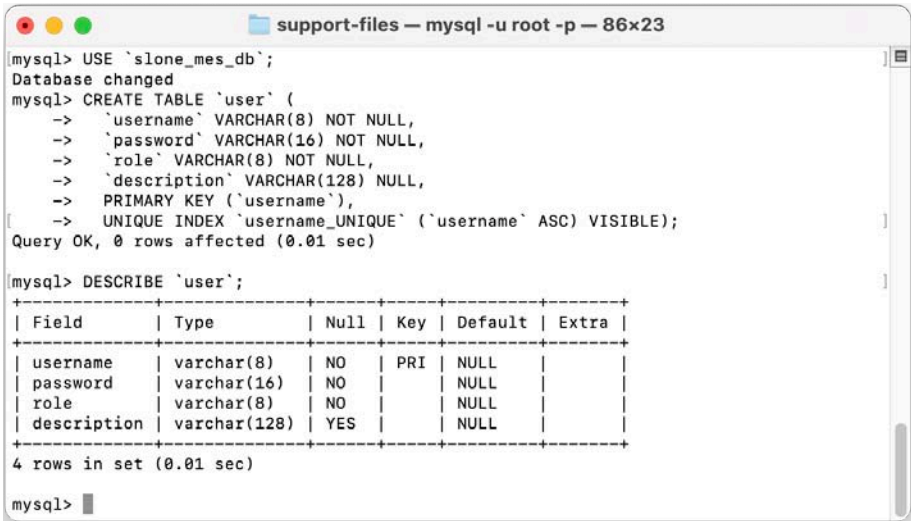


图 3-1　创建用户数据表

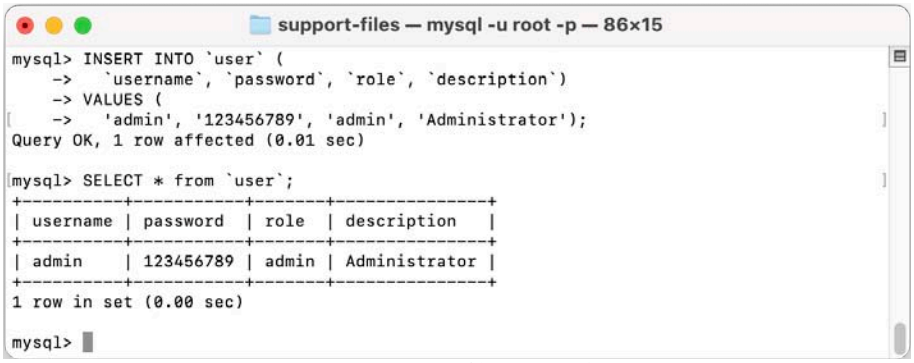


图 3-2　插入管理员帐户

3.3　物料管理数据表的设计

在本课题中，一个物料拥有如表 3-3 所示的属性。要执行的 SQL 语句及执行结果详见附录。

表 3-3　物料管理数据表的设计

属性名	数据类型	完整性约束条件
id	varchar(16)	PK, NN, UQ
name	varchar(32)	NN
quantity_int	int unsigned	NN
quantity_dec	decimal(10,2) unsigned	NN
description	varchar(128)	
isInt	boolean	NN

其中，属性 id 表示一个物料的编号，不超过 16 位英文字符。按照控制要求，一个物料对应一个 id，并且能通过 id 查找到对应的物料，因此把 id 设置为 PK，NN，UQ。

属性 name 表示一个物料的名称，不超过32 位英文字符或 16 个汉字，并且不可省略。

属性 quantity_int 与 quantity_dec 表示物料的数量。若物料的量词为“个”“颗”“张”等个体量词，则把属性 isInt 设置为 true，并使用 quantity_int 存放其数量；若物料的量词为“千克”“厘米”等计量名量词，则把属性 isInt 设置为 false，并使用 quantity_dec 存放其数量。因为数量不为负，所以 quantity_int 与 quantity_dec 均设置为无符号数（unsigned）。属性 quantity_int 为整型（integer，int），占4 个字节，其取值范围为 0 到 4294967295（即 0 到 2^32-1）；属性 quantity_dec 为定点型（decimal），占 12 个字节，其取值范围为 0.00 到 99999999.99；属性 isInt 为布尔型（boolean，bool），占1 个字节，其取值只能为 1（true）或 0（false）。

3.4 设备管理数据表的设计

在本课题中，一个 PLC 拥有如表 3-4 所示的属性。要执行的 SQL 语句及执行结果详见附录。

表 3-4 设备管理数据表的设计

属性名	数据类型	完整性约束条件
name	varchar(32)	NN
IPv4	varchar(15)	PK，NN，UQ
description	varchar(128)	

其中，属性 IPv4 表示一个 PLC 的 IPv4 地址，根据控制要求，一个 IPv4 地址对应一个 PLC，所以把属性 IPv4 设置为 PK，NN，UQ。

3.5 产品管理数据表的设计

在本课题中，一个产品拥有如表 3-5 所示的属性。要执行的 SQL 语句及执行结果详见附录。

表 3-5 产品管理数据表的设计

属性名	数据类型	完整性约束条件
id	varchar(16)	PK, NN, UQ
name	varchar(32)	NN
materials	mediumtext	NN
description	varchar(128)	

其中，属性 materials 表示该产品所需的物料及其数量，其类型为 mediumtext，可存放 16772150 位英语字符。储存的数据由 MES 服务器和 WEB 页面生成和解析。

3.6 订单管理数据表的设计

在本课题中，一个订单拥有如表 3-6 所示的属性。要执行的 SQL 语句及执行结果详见附录。

表 3-6 订单管理数据表的设计

属性名	数据类型	完整性约束条件
id	varchar(16)	PK, NN, UQ
creator	varchar(8)	NN
state	varchar(9)	NN
description	varchar(128)	

product	mediumtext	NN
quantity	int unsigned	NN

其中，属性 creator 表示创建此订单的用户的用户名。

属性 state 表示订单当前的状态，可用的值为 created、queuing、executing、finished、interrupt、failed、deleted 等。

属性 product 表示该订单所生产的产品，其类型为 mediumtext，可存放 16772150 位英语字符。储存的数据由 MES 系统和 WEB 页面生成和解析。

属性 quantity 表示跟订单所生产的产品的数量。

第四章　MES 服务器与共用模块设计

4.1　MES 服务器总体设计

本课题的 MES 服务器采用了如图 4-1 所示的模块化设计，除了包括共用的网络通讯模块、数据加密模块、数据管理模块之外，还包括了生产控制模块等。

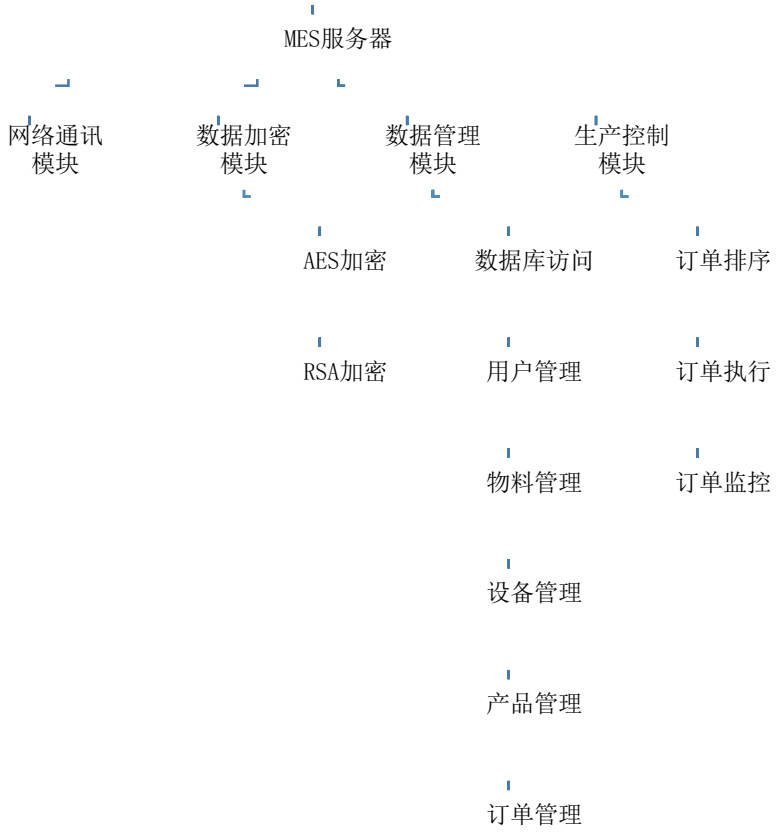


图 4-1 MES 服务器模块化架构

MES 服务器的逻辑图如图 4-2 所示。

其中网络通讯模块用于 MES 服务器与 Web 页面、MES 服务器与现场监控大屏程序和 MES 服务器与现场 PLC 之间的通讯。在 MES 服务器与客户端（包括 Web 页面与现场监控大屏程序，下同）通讯时，收发的信息需使用数据加密模块进行加密。

数据加密模块包括 AES 加密和 RSA 加密。在 MES 服务器与客户端之间首发信息时，均采用 AES 加密，并且加密使用的密钥和初始化向量是由MES 服务器随机生成的。在 MES 服务器与客户端建立连接时，MES 服务器向客户端发送密钥与初始化向量，在断开连接时销毁。发送密钥与初始化向量时，需使用 RSA 加密来保证信息安全。RSA 加密所使用的公钥（public key）和私钥（private key）储存在各自程序目录中的pem 密钥文件中。所有加密后的信息还需经 Base64 编码才可发送。

数据管理模块包括数据库访问以及用户管理、物料管理、设备管理、产品管理、订单管理等功能，除数据库访问功能外的其他功能中的一些类与 Web 页面和现场监控大屏程序共享使用。

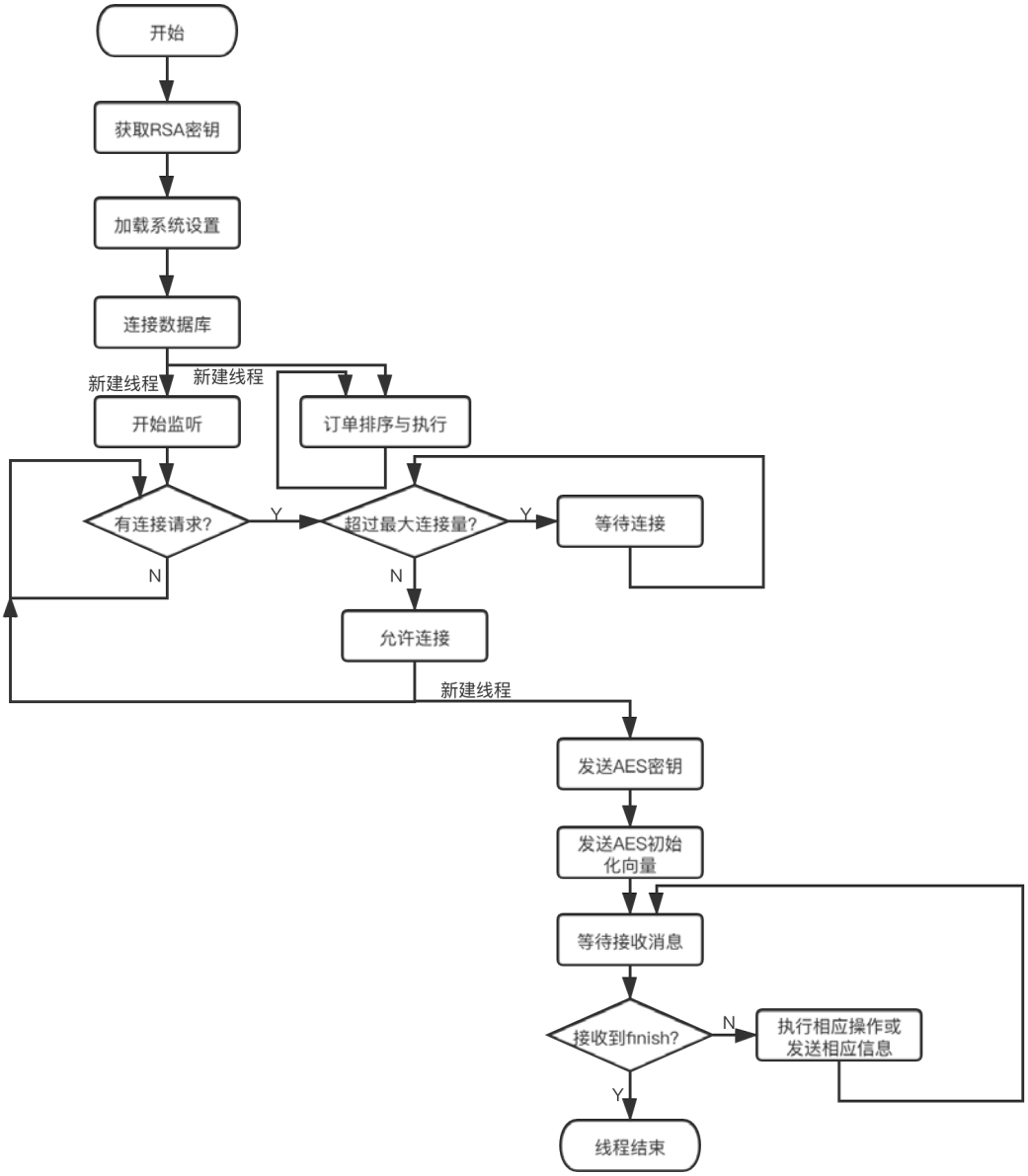


图 4-2　MES 服务器逻辑图

生产控制模块主要包括订单排序、订单执行、订单监控等功能。

4.2　网络通讯模块的开发

在 Visual Studio Community 2019 for Mac 中新建解决方案，命名为 SloneMES，并在该解决方案下新建.NET 类库项目 SloneMES.BSConnector（后文如无特殊说明，新建的项目均在解决方案 SloneMES 下），设置该项目的目标框架为.NET 6.0。

4.2.1　客户端类的开发

在项目 SloneMES.BSConnector 下新建 C#类文件，类名为 Client，并且继承自接口 IDisposable 以支持手动内存回收。该类中包括图 4-3 所示的字段和属性。

其中，TcpClient 类的实例 client 用来建立连接，NetworkStream 类的实例 netStream 为建立网络连接后产生的网络流，BinaryReader 类的实例 br 和 BinaryWriter 类的实例 bw 分别从网络流中读取和写入二进制数据。

```
private readonly TcpClient client;
private readonly NetworkStream netStream;
private readonly BinaryReader br;
private readonly BinaryWriter bw;
private string ReceiveMsg;
private bool IsServer { get; }
```

图 4-3　Client 类中的字段和属性

Client 类中还有发送消息方法 SendMessage()和接收消息方法 ReceiveMesage()，见图 4-4 与图 4-5。

返送消息方法和接收消息方法均有公共的和私有的。在外部调用时均调用公共方法。公共方法又通过开辟新进程来执行私有方法。

Client 类的实例析构时，会自动向 MES 服务器发送 finish 指令，使 MES 服务器同时关闭网络连接，并销毁应用于该实例的 AES 密钥和初始化向量。

```
private void SendMessage(object _sendMsg)
{
    try
    {
        string sendMsg = (string)_sendMsg;
        this.bw.Write(sendMsg);
    }
    catch (Exception e)
    {
        Console.WriteLine("发送失败！");
        Console.WriteLine(e.ToString());
    }
}
public void SendMessage(in string sendMsg)
{
    Console.WriteLine("向{0}发送消息：{1}", this.client.Client.RemoteEndPoint.ToString(), sendMsg);
    Thread sendThread = new(this.SendMessage);
    sendThread.Start(sendMsg);
    sendThread.Join();
}
```

图 4-4　发送消息方法

```
private void ReceiveMessage()
{
    try
    {
        string rcvMsgStr = this.br.ReadString();
        if (rcvMsgStr != null)
        {
            this.ReceiveMsg = rcvMsgStr;
            Console.WriteLine("从{0}接收消息：{1}", this.client.Client.RemoteEndPoint.ToString(), this.ReceiveMsg);
        }
        else throw new("Error 4");
    }
    catch (Exception e)
    {
        Console.WriteLine("接收失败！");
        Console.WriteLine(e.ToString());
        this.ReceiveMsg = null;
        return;
    }
}
public void ReceiveMessage(out string receiveMsg)
{
    Thread receiveThread = new(this.ReceiveMessage);
    receiveThread.Start();
    receiveThread.Join();
    receiveMsg = this.ReceiveMsg;
}
```

图 4-5　接收消息方法

4.2.2 服务器类的开发

在项目 SloneMES.BSConnector 下新建 C#类文件，类名为 Server，并且继承自接口 IDisposable 以支持手动内存回收。该类中包括图 4-6 所示的字段和属性。

```
public int MaxConnector { get; }
private readonly TcpListener listener;
protected readonly Thread acceptThread;
```

图 4-6 Server 类中的字段和属性

其中，TcpListener 类的实例 listener 用来监听网络请求，Thread 类的实例 acceptThread 用来开辟线程以以调用 AcceptClientConnect()方法。

Server 类中有监听并允许接入方法 AcceptClientConnect()和连接并沟通方法 ConnectClient()，见图 4-7 和图 4-8。

AcceptClientConnect()将不断地进行监听，若检测到连接请求并且当前连接的数量不超过设定的最大数量时，与该请求建立连接，若超出最大连接数量则等待。建立连接后，实例化一个 TcpClinet 类，并传给在线程池中新建的执行 ConnectClient()的线程。ConnectClinet()执行时将随机生成一个 AES 加密实例，并且实例化一个 Client 类用于与对方进行通信，当收到 finish 指令后断开连接，断开连接时自动销毁随机生成的 AES 加密实例。

```
private void AcceptClientConnect()
{
    Console.WriteLine("开始监听: {0}", this.listener.Server.LocalEndPoint.ToString());
    while (true)
    {
        try
        {
            TcpClient client = this.listener.AcceptTcpClient();
            if (client != null)
            {
                ThreadPool.QueueUserWorkItem(new(ConnectClient), client);
            }
            else throw new("Error 4");
        }
        catch (Exception e)
        {
            Console.WriteLine("连接客户端失败! ");
            Console.WriteLine(e.ToString());
            continue;
        }
    }
}
```

图 4-7 监听并允许接入方法

```
private void ConnectClient(object _client)
{
    using AES aes = new();
    using TcpClient client = (TcpClient)_client;
    using Client myClient = new(client, true);
    myClient.SendMessage(aes.GetKey());
    myClient.SendMessage(aes.GetIV());
    while (true)
    {
        string cmdStr;
        myClient.ReceiveMessage(out string cipherText);
        if (cipherText != null && cipherText.Length > 0)
        {
            cmdStr = aes.Decrypt(cipherText);
        }
        else break;

        if (cmdStr != "finish")
        {
            myClient.SendMessage(aes.Encrypt(this.Answer(cmdStr)));
        }
        else break;
    }
}
```

图 4-8 连接并沟通方法。

4.3 数据管理模块的开发

下文将介绍用户类、产品类和订单类的设计与开发，物料类、PLC 类等类的代码与之类似，详见源代码。

4.3.1 用户类的开发

新建.NET 类库项目 SloneMES.Managements，设置该项目的目标框架为.NET 6.0。在该项目中新建 C#类文件，类名为 User，并使用关键字 record 代替 class 进行声明。User 类的属性、构造函数、解构函数见附录。

在 User 类中重写了转字符串方法 ToString()，如图 4-9 所示。

在 User 类中还有 2 个静态方法，分别是从字符串构造实例方法 FromString()和验证账户密码是否符合规范的方法 IsRegular()，如图 4-10 所示。

```
public override string ToString() =>
    $"{this.Username.Length}username:{this.Username};" +
    $"{this.Password.Length:D2}password:{this.Password};" + "" +
    $"{this.Role.ToString().Length}role:{this.Role};" +
    $"{this.Description.Length:D3}description:{this.Description}.";
```

图 4-9 User 类中的 ToString()方法

```
public static User FromString(in string userStr)
{
    string str = new(userStr);
    if (str == null || str.Length <= 0) return new("用户字符串错误");

    if (!str.EndsWith(".")) return new("用户字符串错误");
    else str = str[0..^1];

    string username;
    if (int.TryParse(str.Substring(0, 1), out int nameLength) && str.Substring(1, 9) == "username:")
    {
        if (nameLength < 4 || nameLength > 8) return new("帐户长度错误");
        else username = str.Substring(10, nameLength);
    }
    else return new("帐户字符串错误");

    if (str.Substring(10 + nameLength, 1) == ";")
        str = str[(11 + nameLength)..];
    else return new("用户字符串错误");

    string password;
    if (int.TryParse(str.Substring(0, 2), out int pwdLength) && str.Substring(2, 9) == "password:")
    {
        if (pwdLength < 8 || pwdLength > 16) return new("密码长度错误");
        else password = str.Substring(11, pwdLength);
    }
    else return new("密码字符串错误");

    if (str.Substring(11 + pwdLength, 1) == ";")
        str = str[(12 + pwdLength)..];
    else return new("用户字符串错误");

    if (!IsRegular(username) || !IsRegular(password)) return new("用户名或密码中含有其他字符");

    Roles role;
    if (int.TryParse(str.Substring(0, 1), out int roleLength) && str.Substring(1, 5) == "role:")
        role = Enum.Parse<Roles>(str.Substring(6, roleLength));
    else return new("权限字符串请求错误");

    if (str.Substring(6 + roleLength, 1) == ";")
        str = str[(7 + roleLength)..];
    else return new("用户字符串错误");

    string description;
    if (int.TryParse(str.Substring(0, 3), out _) && str.Substring(3, 12) == "description:")
        description = str[15..];
    else return new("备注字符串错误");

    return new(username, password, role, description);
}
public static bool IsRegular(string s) => Regex.IsMatch(s, "^[a-zA-Z0-9]*$");
```

图 4-10 User 类中的静态方法

其中，FromString()方法将一步步地解析由 ToString()方法生成的字符串，生成一个 User 实例并返回。FromString()和 ToString()搭配使用，即可在网络中传输 User 实例。

IsRegular()方法将检查账户和密码是否符合规范，采用了正则表达式（Regular Expression，Regex）的验证方法。IsRegular()方法调用了 Regex 类中的 IsMatch()方法，其中匹配模式（pattern）为“^[a-zA-Z0-9]*$”，含义为从字符串开头到字符串结尾只能包括小写字母、大写字母以及数字。Regex 类由.NET 平台提供，在命名空间 System.Text.RegularExpressions 下。

在 User 类中还有用来表示用户权限的枚举和静态类，详见附录。

4.3.2 产品类的开发

在 SloneMES.Managements 项目中新建 C#类文件，类名为 Product，并使用关键字 record 代替 class 进行声明。Product 类的属性跟数据库中的有些出入，如图 4-11。

在数据库中，属性 materials 的数据类型为 mediumtext，在网络通信中收发的数据类型为 string，但在程序中，属性Materials 的数据类型为 Material 类的列表 List<Material>，在把 Product 类的实例存放进数据库以及网路发送时需要把属性 Materials 转换为字符串，在从数据库中读取以及网络接收 Product 实例时也需要做与之相反的转换。用于转换的代码在方法 ToString()和 FromString()中有所体现。

在 ToString()方法中遍历了数据类型为列表 List<Material>的属性 Materials，并调用了该属性中的物料实例的 ToString()方法，并将生成的字符串整合进临时变量materialsStr 中，如图 4-12 所示。

```
public string ID { get; }
public string Name { get; }
public List<Material> Materials { get; }
public string Description { get; }
```

图 4-11　Product 类的属性

```
public override string ToString()
{
    string materialsStr = "";
    foreach (Material material in this.Materials)
    {
        materialsStr += $"{material.ToString().Length:D3}{material}";
    }
    return
        $"{this.ID.Length:D2}id:{this.ID};" +
        $"{this.Name.Length:D2}name:{this.Name};" +
        $"{this.Description.Length:D3}description:{this.Description};" +
        $"materials:{materialsStr}.";
}
```

图 4-12　Product 类中的 ToString()方法

在 FromString()方法中，循环拆分物料列表的字符串，并将拆分到的字符串片段依次传入物料类的 FromString()方法以得到物料实例，再将物料实例依次添加到数据类型为列表 List<Material>的临时变量 materials 中，如图 4-13 所示。

```
public static Product FromString(in string productStr)
{
    string str = new(productStr);
    if (str == null || str.Length <= 0) return new("产品字符串错误");

    if (!str.EndsWith(".")) return new("产品字符串错误");
    else str = str[0..^1];

    string id;
    if (int.TryParse(str.Substring(0, 2), out int idLength) && str.Substring(2, 3) == "id:")
    {
        if (idLength < 1 || idLength > 16) return new("编号长度错误");
        else id = str.Substring(5, idLength);
    }
    else return new("编号字符串错误");

    if (str.Substring(5 + idLength, 1) == ";")
        str = str[(6 + idLength)..];
    else return new("产品字符串错误");

    string name;
    if (int.TryParse(str.Substring(0, 2), out int nameLength) && str.Substring(2, 5) == "name:")
    {
        if (nameLength < 1 || nameLength > 32) return new("名称长度错误");
        else name = str.Substring(7, nameLength);
    }
    else return new("名称字符串错误");

    if (str.Substring(7 + nameLength, 1) == ";")
        str = str[(8 + nameLength)..];
    else return new("产品字符串错误");

    string description;
    if (int.TryParse(str.Substring(0, 3), out int dctLength) && (str.Substring(3, 12) == "description:"))
        description = str.Substring(15, dctLength);
    else return new("备注字符串错误");

    if (str.Substring(15 + dctLength, 1) == ";")
        str = str[(16 + dctLength)..];
    else return new("产品字符串错误");

    List<Material> materials = new();
    if (str.Substring(0, 10) == "materials:")
    {
        str = str[10..];
        while (str != "")
        {
            if (int.TryParse(str.Substring(0, 3), out int matLength))
            {
                materials.Add(Material.FromString(str.Substring(3, matLength)));
                str = str[(3 + matLength)..];
                continue;
            }
            else return new("原料字符串错误");
        }
    }
    else return new("原料字符串错误");

    return new Product(in id, in name, in materials, in description);
}
```

图 4-13　Product 类中的 FromString()方法

4.3.3　订单类的开发

在 SloneMES.Managements 项目中新建 C#类文件，类名为 Order，并使用关键字 record 代替 class 进行声明。

Order 类中包括一个委托和一个该委托的事件，如图 4-14 所示。的事件 OnStateChanged 将在订单的状态发生变化时调用。由于需要访问并修改数据库，并且该类所在项目并不引用项目 SloneMES.DBConnector，所以事件调用的具体方法在项目 SloneMES.Server 中的 SloneMES_Server 类中定义并绑定，详见下文。

在 Order 类中还有用来表示订单状态的枚举，详见附录。

```
public delegate void StateChangedHandler(in string id, in OrderState state);
private event StateChangedHandler? OnStateChanged;
```

图 4-14　Order 类的委托与事件

4.4　生产控制模块的开发

在 SloneMES.Managements 项目中新建 C#类文件，类名为 Process1，并且将该类设置为静态类。Process1 类中最重要的方法即为制作订单的方法 Make()，该方法的设计流程如图 4-15 所示，代码如图 4-16 所示。Make()方法在线程池中以后台线程的方式执行，执行的订单的数量由 Process1 的静态字段 OrderAmount 表示，Make()方法开始执行时使 OrderAmount 的值加 1，Make()方法退出时使 OrderAmount 的值减 1。

在该方法中，首先检查静态类 Process1 中是否设置了所需的 PLC，若没有设置则退出执行，若以设置将锁定 PLC 以免被其他方法修改导致订单执行失败。然后再检查传进来的订单实例的状态是否为 queuing，若不是queuing 的话则退出执行，若为queuing 则开始执行并将状态修改为 executing。接着则进入 for 循环，直到制作的数量达到订单上要求的数量后结束循环，否则将一直制作产品。在该 for 循环中，首先检查各个物料的数量是否满足制作一件产品的数量，若不满足则等待。

订单执行时，根据 PLC 向 MES 发来的数据回复相应的数据以控制生产线的运行，此部分代码需与 PLC 程序互相配合。生成 PLC 控制数据的方法在 PLC 类中，为静态方法，代码详见源代码。在制作完一件产品后再次检查订单状态，若果订单在制作期间被其他方法修改状态为 interrupt 等，则停止执行。并且，在制作完一件产品后需要在数据库中修改其使用的物料的数量。

查询物料数量的静态事件 GetMaterialQuantityInt、GetMaterialQuantityDec 以及修改物料数量的静态事件 OnReduceMaterialsQuantity 所调用的方法均在项目 SloneMES.Server 中的 SloneMES_Server 类中定义并绑定，详见下文。

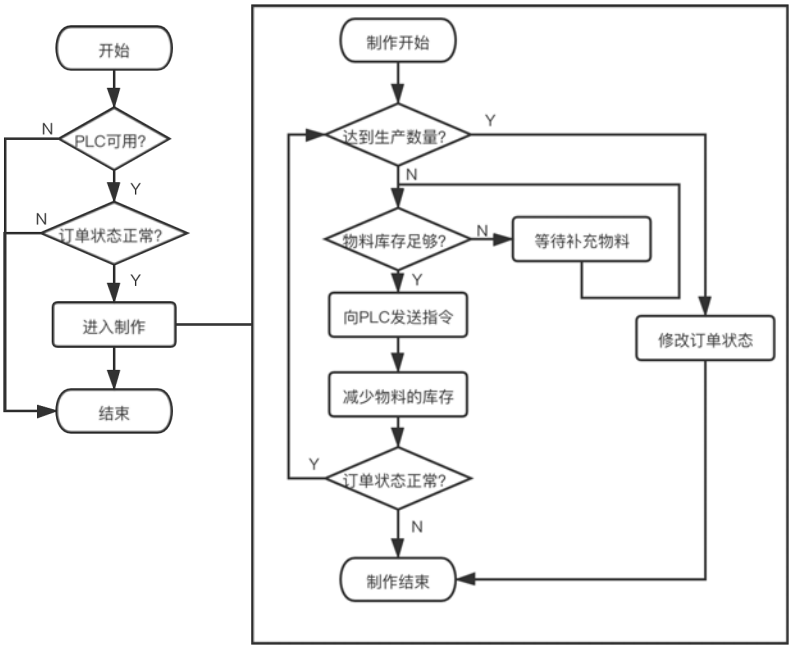


图 4-15　Make()方法流程图

```
private static void Make(Order order)
{
    if (PLCs == null) return; //检查是否设置了PLC
    lock (PLCs)
    {
        if (order == null || order.State != Order.OrderState.queuing) return; //检查订单状态是否为queuing
        try
        {
            order.ChangeState(Order.OrderState.executing); //修改订单状态为executing
            OrderAmount++; //正在执行的订单数量+1

            //初始化一些变量
            Bytes oNo = new(Encoding.ASCII.GetBytes(order.ID));
            Bytes oPos = PLC.Zeros(4);
            Bytes pNo = new(Encoding.ASCII.GetBytes(order.Product.ID));

            //开始执行，直到达成所需数量
            for (uint quantity = 0u; quantity < order.Quantity; quantity++)
            {
                //检查物料数量是否足够，不足则等待
                bool isQuantityEnough;
                do
                {
                    isQuantityEnough = true;
                    foreach (Material material in order.Product.Materials)
                    {
                        uint quantity_int = 0u;
                        decimal quantity_dec = 0m;
                        if (material.IsInt)
                            quantity_int = GetMaterialQuantityInt(material.ID);
                        else
                            quantity_dec = GetMaterialQuantityDec(material.ID);

                        if (material.IsInt)
                        {
                            if (quantity_int < material.Quantity_int)
                                isQuantityEnough = false;
                        }
                        else
                        {
                            if (quantity_dec < material.Quantity_dec)
                                isQuantityEnough = false;
                        }
                    }
                }
                while (!isQuantityEnough);

                //依次向每个工作站发送控制命令
                for (int i = 0; i < PLCs.Count; i++)
                {
                    while (order.State == Order.OrderState.executing)
                    {
                        Dictionary<string, Bytes> command = PLCs[i].GetCommand(); //接收PLC发来的数据
                        if (PLC.IsEqual(command["TcpIdent"], PLC.tcpIdentServer)) //检查接收头是否匹配
                        {
                            //初始化一些变量
                            Bytes mClass = command["mClass"];
                            Bytes mNo = command["mNo"];
                            Bytes requestID = command["RequestID"];
                            Bytes resourceID = command["ResourceID"];
                            Bytes command2 = null;

                            if (PLC.IsEqual(mClass, new() { 0x00, 0x64 })) //mClass = 0064
                            {
                                if (PLC.IsEqual(mNo, new() { 0x00, 0x04 })) //mNo = 0004
                                    command2 = PLC.GetFirstOpForRsc(requestID, resourceID, oNo, oPos, pNo);
                                else if (PLC.IsEqual(mNo, new() { 0x00, 0x06 })) //mNo = 0006
                                    command2 = PLC.GetOpForONoOPos(requestID, resourceID, oNo, oPos, pNo);
                                else if (PLC.IsEqual(mNo, new() { 0x00, 0x6F })) //mNo = 006F
                                    command2 = PLC.GetFreeString(requestID, resourceID, oNo, oPos, pNo);
                                else throw new("功能编号2错误！");
                            }
                            else if (PLC.IsEqual(mClass, new() { 0x00, 0x65 })) //mClass = 0065
                            {
                                if (PLC.IsEqual(mNo, new() { 0x00, 0x14 })) //mNo = 0014
                                {
                                    PLCs[i].SetCommand(PLC.OpEnd(requestID, resourceID));
                                    break;
                                }
                                else if (PLC.IsEqual(mNo, new() { 0x00, 0x0A })) //mNo = 000A
                                    command2 = PLC.OpStart(requestID, resourceID, oNo, oPos, pNo);
                                else if (PLC.IsEqual(mNo, new() { 0x00, 0x01 })) //mNo = 0001
                                    command2 = PLC.SetPar(requestID, resourceID, oNo, oPos, pNo);
                                else if (PLC.IsEqual(mNo, new() { 0x00, 0x0F })) //mNo = 000F
                                {
                                    PLCs[i].SetCommand(PLC.OpReset(requestID, resourceID));
                                    break;
                                }
                                else throw new("功能编号2错误！");
                            }
                            else throw new("功能编号1错误！");

                            PLCs[i].SetCommand(command2);
                            continue;
                        }
                        else throw new("接收字错误！");
                    }
                    if (order.State != Order.OrderState.executing) throw new OrderStateError("订单状态错误！");
                }
                OnReduceMaterialsQuantity(order.Product.Materials); //减少物料数量
            }
            order.ChangeState(Order.OrderState.finished); //修改订单状态为finished
        }
        catch (OrderStateError e)
        {
            Console.WriteLine($"Error:{e}");
        }
        catch (Exception e)
        {
            order.ChangeState(Order.OrderState.failed);
            Console.WriteLine($"Error:{e}");
        }
        finally
        {
            OrderAmount--;
        }
    }
}
```

图 4-16　Process1 类中的 Make()方法

4.5 MES 服务器的综合开发

新建.NET 控制台应用程序项目 SloneMES.Server，设置该项目的目标框架为.NET 6.0。新建 C#类文件，类名为 MESServer，MESServer 类中重写了 BSConnector.Server 类中的 Answer()方法，如图 4-17 所示。

```
protected sealed override string Answer(in string cmdStr)
{
    if (cmdStr.Length < 8) return "Error###请求错误！";
    string operation = cmdStr.Substring(0, 8).Replace("#", "");
    string param = cmdStr[8..];
    try
    {
        return operation switch
        {
            //用户管理
            "Login" => this.Login(param),
            "Logout" => this.Logout(param),
            "AddUser" => this.AddUser(param),
            "DelUser" => this.DelUser(param),
            "GetUsers" => this.GetUsers(param),

            //设备管理
            "CntToPLC" => this.ConnectToPLC(param),
            "DcntPLC" => this.DisconnectFromPLC(param),
            "AddPLC" => this.AddPLC(param),
            "DelPLC" => this.DelPLC(param),
            "GetPLCs" => this.GetPLCs(param),
            "PLCState" => this.GetPLCState(param),

            //物料管理
            "AddMat" => this.AddMaterial(param),
            "DelMat" => this.DelMaterial(param),
            "ChgMat" => this.ChangeMaterial(param),
            "GetMats" => this.GetMaterials(param),

            //产品管理
            "AProduct" => this.AddProduct(param),
            "DProduct" => this.DelProduct(param),
            "GProduct" => this.GetProducts(param),

            //流程管理
            "Set1PLCs" => this.SetProcess1PLCs(param),
            "Get1PLCs" => this.GetProcess1PLCs(param),

            //订单管理
            "AddOrder" => this.AddOrder(param),
            "DelOrder" => this.DelOrder(param),
            "ApvOrder" => this.ApproveOrder(param),
            "IntOrder" => this.InterruptOrder(param),
            "GetOrder" => this.GetOrders(param),

            //未识别的指令
            _ => "Error###指令错误！",
        };
    }
    catch (Exception e)
    {
        return $"Error###{e}";
    }
}
```

图 4-17 MESServer 类中的 Answer()方法

该方法接收的参数是由 Web 页面或现场监控大屏程序通过网络发送来的字符串。其中字符串的前 8 位为指令，不足 8 位的用“#”填充，8 位后的为参数，MES 服务器根据指令调用相应的方法来实现具体功能，若操作成功，则返回“Success#”或特定用途的字符串；若操作失败则返回以“Error###”开头的表示错误信息的字符串。其中，可供请求的功能有登录，登出，用户的添加、删除、获取，PLC 的连接、断开、添加、删除、获取，物料的添加、删除、修改数量、获取，产品的添加、删除、获取，设置生产流程 1 的设备，获取生产流程 1 的设备，订单的添加、删除、批准、终止、获取等。

MESServer 类中定义着供上述事件绑定的方法 ChangeOrderState()、GetMaterialQuantityInt()、GetMaterialQuantityDec()、ReduceMaterialsQuantity()方法，代码见附录。

第五章　Web 页面与现场监控大屏程序设计

5.1　Web 页面设计

5.1.1　总体设计

Web 页面的架构如图 5-1 所示，其中多数模块与 MES 服务器共用。新建 Blazor Server 项目 SloneMES.Web，设置该项目的目标框架为.NET 6.0，本节中的所有开发均在该项目中进行。Web 页面仅用来对 MES 服务器进行用户界面展示，并不直接对生产线进行控制与监视。

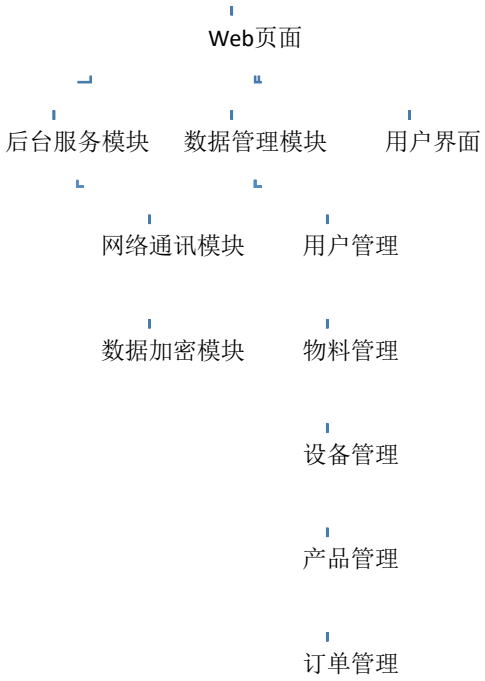


图 5-1　WEB 页面架构

电脑端的页面总体可分为 2 部分，左侧有一个导航栏，右侧是具体的页面。在移动端，导航栏将被折叠并置于上方，已更多的现实页面内容。其中，导航栏中显示的连接将根据是否登录以及用户权限的不同而变化。

主页

图 5-2　WEB 页面设计效果草图（电脑端）

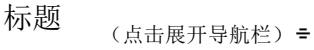


图 5-3　WEB 页面设计效果草图（移动端）

5.1.2　主页设计

主页主要用来进行用户登录和注销的操作。在未登录时，主页将有2 个输入框以供用户输入用户名和密码，并提供登录按钮，并且在用户按下回车时也能进行登录操作，若登录失败，会有相关提示。在登录后，页面上将显示用户名，并提供一个注销按钮以供用户注销。并且，当用户在任意页面上关闭，将会自动注销。主页的设计的效果如图 5-4 与图 5-5 所示（仅展示电脑端效果，移动端效果类似，下同）。

图 5-4　主页设计效果（登录前）

图 5-5　主页设计效果（登录后）

5.1.3 用户管理页面设计

在登录后并且用户的权限为 admin 时，导航栏才会显示用户管理页面的链接，点击链接后跳转到用户管理页面。在用户管理页面展示了数据库里所有的用户，并且显示该用户是否已经登录。在用户列表下方有 3 个按钮，其功能分别为新增用户、删除用户和刷新用户数据。

当点击新增用户时，页面会弹出一个窗口，在弹出窗口上输入用户名、密码、权限、备注等相关信息即可向 MES 服务器发出新增用户的命令。

当点击删除按钮后，再次点击用户名即可弹出确认框，点击确认后会向 MES 服务器发出删除用户的命令。但若要删除 admin 帐户、非 admin 管理员帐户删除其他管理员帐户、删除正在登录的帐户则不会发出相关请求或者不会被执行相关指令。

当点击刷新时，页面将再次从 MES 服务器获取用户信息。

执行是否成功都会有相应的提示。用户管理页面的设计的效果如图 5-6、图 5-7 与图 5-8 所示，代码见附录，其他页面的代码也与此页面类似。

图 5-6 用户管理页面设计效果

图 5-7 用户管理页面设计效果（删除用户时）

图 5-8 用户管理页面设计效果（新增用户时）

5.1.4 设备管理页面设计

在登录后并且用户的权限为 admin 时，导航栏才会显示设备管理页面的链接，点击链接后跳转到设备管理页面。设备管理页面总体上与用户管理页面相似，增加了连接 PLC 以及与 PLC 断开连接的按钮。设备管理页面的设计效果如图 5-9 所示。

图 5-9 设备管理页面设计效果

其中，设备状态由页面调用后台服务模块中的获取设备状态方法 GetPLCState()，该方法向 MES 服务器发送控制命令“PLCState”并附上 PLC 的 IPv4 地址作为参数，MES 服务器收到后从 PLC 获取状态数据并通过网络发送给 Web 页面解析。Web 页面收到状态数据后通过调用 PLC 类中的 PLCState 子类的构造函数来生成 PLCState 类的实例，并根据PLCState 实例中的属性来生成要展示的状态字符串，例如，如果PLCState 实例中的布尔属性 Error0 为真（true），则显示“错误0”。后台服务模块中的获取设备状态方法 GetPLCState()如图 5-10 所示。

```
public string GetPLCState(in string ipv4)
{
    string cmdStr = "PLCState" + ipv4;
    this.SendMessage(cmdStr);
    string anwser = this.ReceiveMessage();

    string addState = anwser.Substring(0, 8).Replace("#", "");
    if (addState == "Error") return anwser[8..];
    else if (addState == "State")
    {
        string @return = "";
        string stateStr = anwser[8..];
        PLC.PLCState state = PLC.PLCState.FromString(stateStr);
        if (state.MESMode) @return += "MES模式;";
        if (state.Error0) @return += "错误0;";
        if (state.Error1) @return += "错误1;";
        if (state.Error2) @return += "错误2;";
        if (state.Resetting) @return += "复位;";
        if (state.Busy) @return += "繁忙;";
        if (state.Manual) @return += "手动;";
        if (state.Automatic) @return += "自动;";
        if (@return != "") @return = @return[0..^1];
        return @return;
    }
    else return "未知状态";
}
```

图 5-10 后台服务模块中的 GetPLCState()方法

PLCState 子类中构造函数如图 5-11 所示。该构造函数需要传入一个 Bytes 类型的参数（Bytes 类型为 List<byte>的缩写，即以字节类型构成的列表），并且要求其长度为4。解析状态的代码见图中红框，其中 Convert.ToBoolean()方法把整数转化成布尔数，整表达式的意义就是取出该字节的第 i 位并转化成布尔类型，Convert.ToBoolean()方法的参数即

为从该字节中取出的某一位的数。其中，运算符“<<”为左移位运算符，表达式“1<<n”即为 2 的 n 次方（n 为非负整数），例如，1=0b00000001，1<<3=0b00001000=8=2^3。但是，对于取第 0 位的数来说，一个 8 位的 2 进制数除以 2 的 7 次方后只剩第 0 位的数了，所以不再需要对商进行除以 2 取余的操作；与之对应的，对于取第 7 位的数来数，一个8 位的 2 进制数直接除以 2 取余即可得到第 7 位的数，无需先除以 2 的 0 次方再除以 2 取余（因为原数除以 2 的 0 次方等于原数）。

```
public PLCState(in Bytes state)
{
    if (state.Count == 4)
        this.state = state;
    else
        this.state = Zeros(4);
    this.ID = state[0] * byte.MaxValue + this.state[1];
    this.Type = (PLCType)this.state[2];
    this.MESMode = Convert.ToBoolean(this.state[3] / (1 << 7));
    this.Error0 = Convert.ToBoolean(this.state[3] / (1 << 6) % 2);
    this.Error1 = Convert.ToBoolean(this.state[3] / (1 << 5) % 2);
    this.Error2 = Convert.ToBoolean(this.state[3] / (1 << 4) % 2);
    this.Resetting = Convert.ToBoolean(this.state[3] / (1 << 3) % 2);
    this.Busy = Convert.ToBoolean(this.state[3] / (1 << 2) % 2);
    this.Manual = Convert.ToBoolean(this.state[3] / (1 << 1) % 2);
    this.Automatic = Convert.ToBoolean(this.state[3] % 2);
}
```

图 5-11　PLCState 子类中的构造函数

5.1.5　物料管理页面设计

在登录后并且用户的权限为 admin 时，导航栏才会显示物料管理页面的链接，点击链接后跳转到物料管理页面。物料管理页面在总体上与用户管理页面、设备管理页面相似，只不过可以直接在表格中修改相应物料的数量。物料管理页面的设计效果如图5-12 所示。

图 5-12　物料管理页面设计效果

5.1.6　产品管理页面设计

在登录后并且用户的权限为 admin 时，导航栏才会显示产品管理页面的链接，点击链接后跳转到产品管理页面。产品管理页面在总体上与上述页面相似，但在表格中不直接显示每个产品所使用的物料，而是显示一个按钮，按下时弹出一个弹出框，并在弹出框中显示所用的物料。新增产品时，可以从已有的物料中选取要使用的物料，并设置用量；若设置的用量为 0，则表示该物料是可选的，并且应在建立订单时确定其具体用量，体现了可定制化的生产产品。产品管理页面的设计效果如图 5-13、5-14 与 5-15 所示。

图 5-13　产品管理页面设计效果

图 5-14　产品管理页面设计效果（查看产品所用物料）

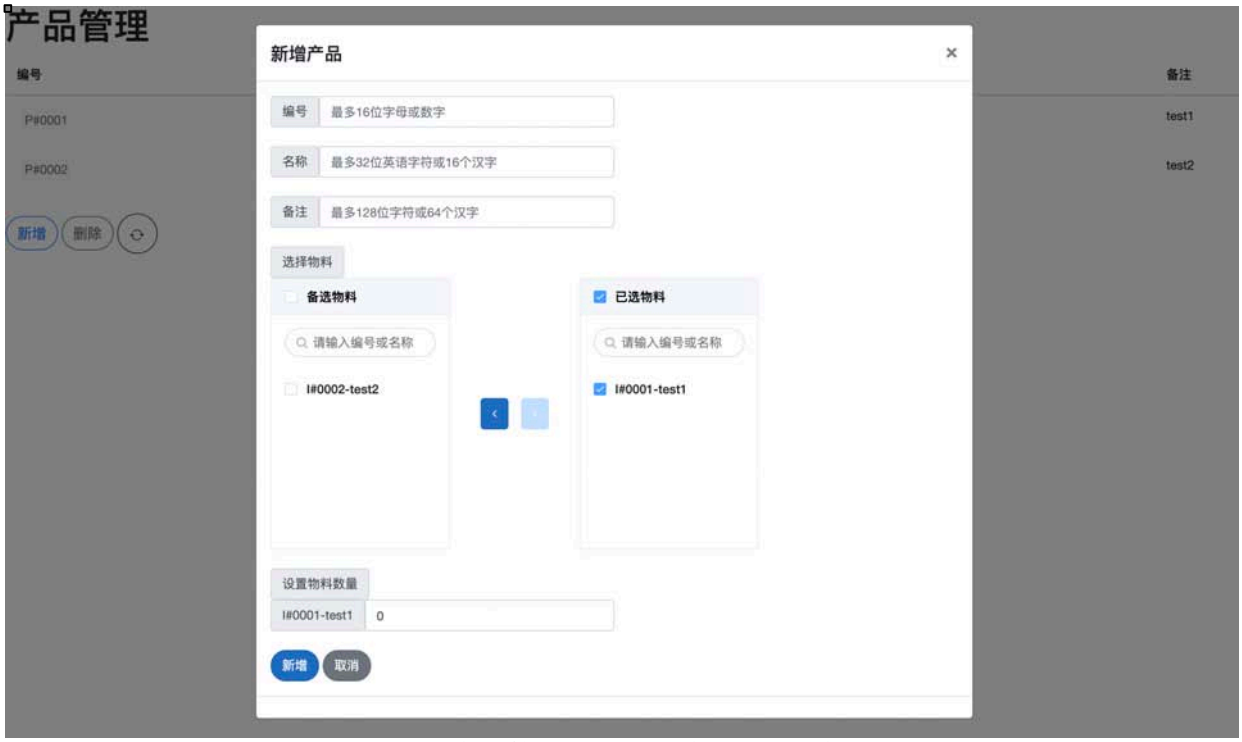


图 5-15　产品管理页面设计效果（新增产品）

5.1.7　订单管理页面设计

在登录后并且用户的权限为 admin 时，导航栏才会显示订单管理页面的链接，点击链接后跳转到订单管理页面。订单管理页面在总体上与上述页面相似，但在表格中不直接显示每个订单所生产的产品，而是显示一个按钮，按下时弹出一个弹出框，并在弹出框中显示生产的产品及该产品的物料信息。新增订单时，可以从已有的产品中选取要生产的产品，并设置产量；若该产品中有可选物料，则应该明确其用量。

此外，还可以对订单进行批准或取消的操作，点击批准按钮会将该订单的状态设为 queuing，点击取消按钮会将该订单的状态设为 interrupt，已经完成的订单除外。订单管理页面的设计效果如图 5-16、5-17 与 5-18 所示。

图 5-16　订单管理页面设计效果

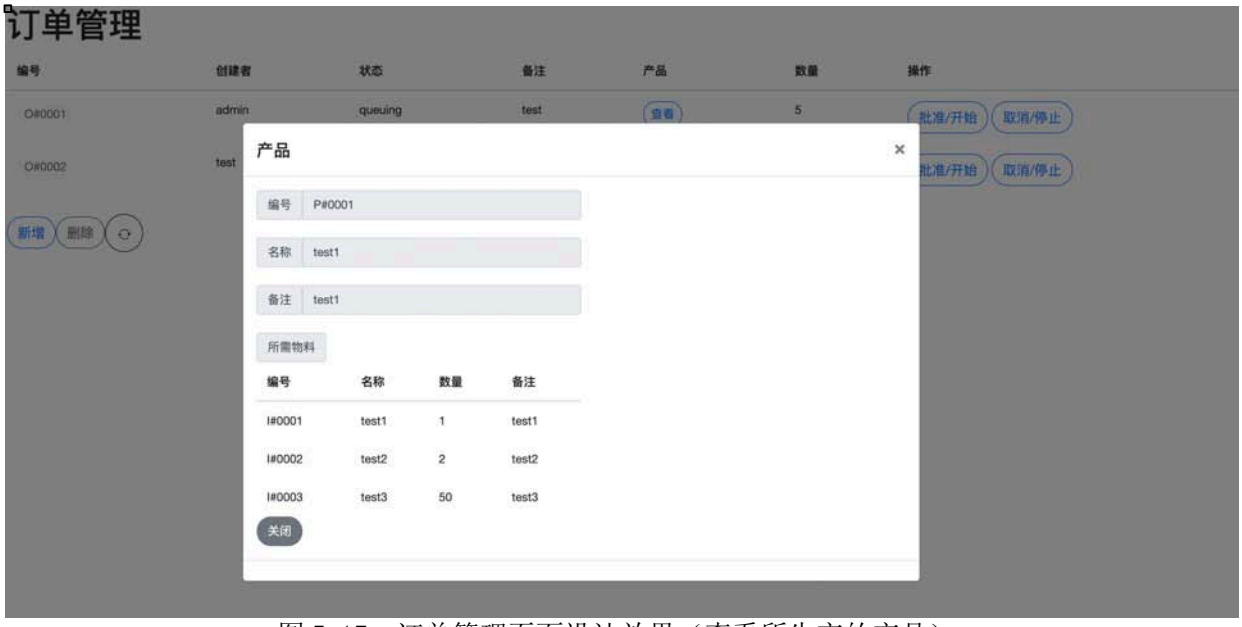


图 5-17　订单管理页面设计效果（查看所生产的产品）

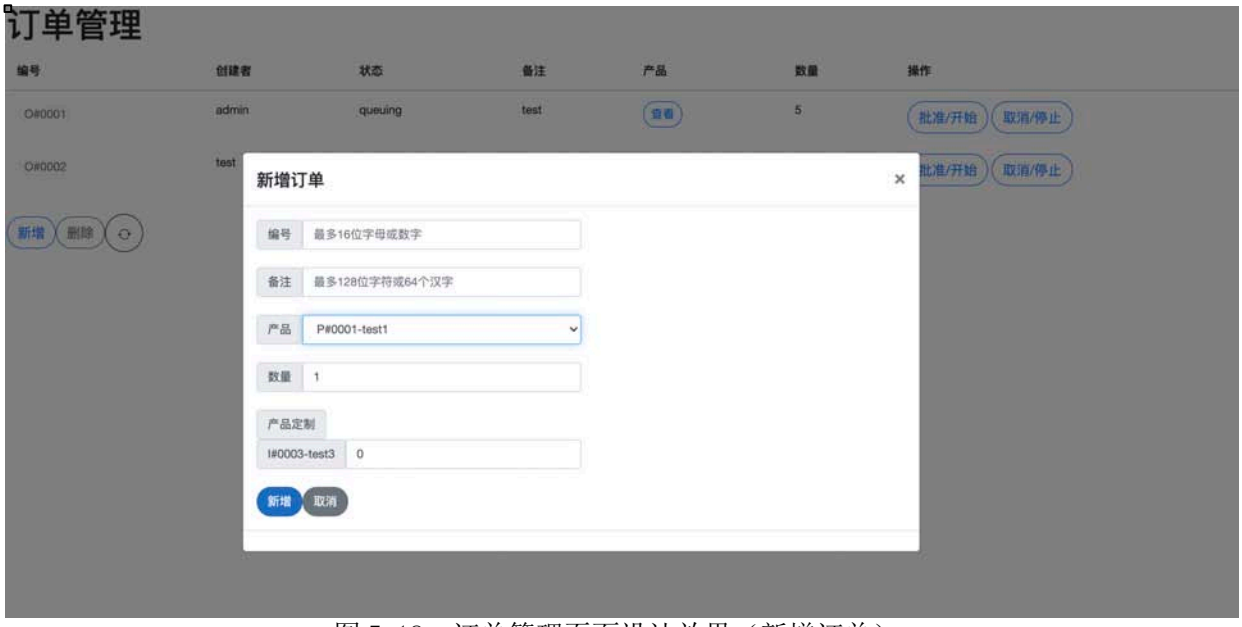


图 5-18　订单管理页面设计效果（新增订单）

5.1.8 其他页面设计

在登录后并且用户的权限不为 admin 时，导航栏才会显示历史订单页面和新增订单页面的链接，点击链接后跳转到相应页面。

历史订单页面与订单管理页面类似，但是只能查看当前用户所创建的订单，也不能对订单进行删除、批准和取消。

新增订单页面与订单管理页面的新增订单弹窗类似，不过新增订单时若用户权限不为 admin 则订单的状态为 created，需经管理员批准后才会加入订单队列并等待执行；若用户权限为 admin，则订单的状态为 queuing，无需批准即可等待执行。

历史订单页面、新增订单页面的设计效果如图 5-19 与 5-20 所示。

图 5-19　历史订单页面设计效果

图 5-20　新增订单页面设计效果

5.2　现场监控大屏程序设计

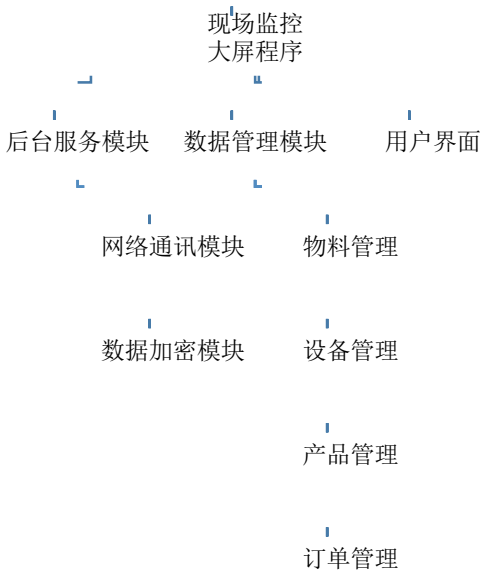


图 5-21　现场监控大屏程序的架构

现场监控大屏程序的架构如图 5-21 所示，其中多数模块与 MES 服务器和 Web 页面共用。新建 Blazor Server 项目 SloneMES.Monitor，设置该项目的目标框架为.NET 6.0，本节中的所有开发均在该项目中进行。

现场监控大屏程序主要在装有 Windows、macOS、Linux 系统的生产线上位机上运行，其页面的设计效果与 Web 页面相似，功能与 Web 页面相比有所删减。

在该项目的目录中打开终端，并执行“*dotnet tool install ElectronNET.CLI -g*”来安装 Electron.NET 的命令行工具。安装成功后执行“*electronize init*”来使项目做好打包称为独立的应用程序的初始化工作，如图 5-22 所示。

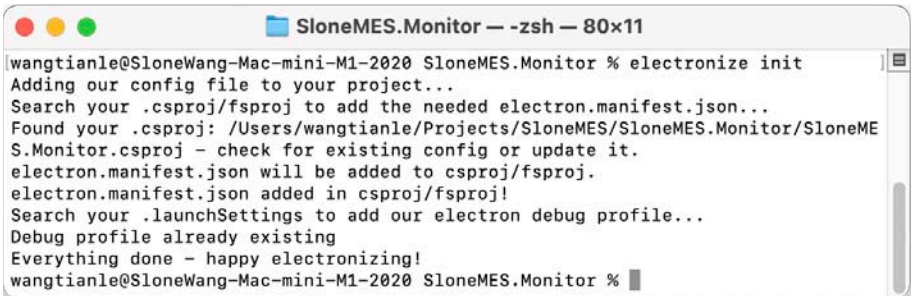


图 5-22　初始化项目 SloneMES.Monitor

第六章　MES 系统的调试及总结

6.1　自动编译脚本的编写

在解决方案的目录中新建目录 Publish，并在 Publish 目录中新建 Python 文件 publish.py 并使用 Visual Studio Code 进行编写，代码见附录。

其中最主要的函数为 publish()函数。该函数接收2 个参数，其中第一个参数为表示系统及架构的字符串，第二个参数为项目名。该函数根据这两个参数可生成在 Windows、macOS、Linux 下都通用的命令行指令，对于除 SloneMES.Monitor 以外的项目将会调用 os.system()函数来执行该指令来进行编译。由于项目 SloneMES.Monitor 使用了 Electron.Net 并需要进一步打包，所以本自动编译脚本对该项目不作编译，仅生成并显示编译命令，需在项目所在目录中手动执行，并手动将打包好的程序复制到 Publish 文件夹中。

在终端中执行脚时时要传入 1 到 2 个参数，其中第 1 个参数为项目的名称，可在 SloneMES.Server、SloneMES.Web、SloneMES.Monitor 中选择，也可输入all 编译全部项目的全部受支持的目标系统及架构。第 2 个参数为可选参数，当 1 个参数为 all 时不能给出第 2 个参数；当第 1 个参数为项目名时，第 2 个参数为要编译的目标系统及架构，或者省略以编译跨平台版本。脚本运行效果如图 6-1 所示。

图 6-1　自动编译脚本的运行

6.2　MES 系统的调试

6.2.1　项目编译及本机调试

在 Publish 目录中打开终端，并依次执行“*./publish.py SloneMES.Server*”与“*./publish.py SloneMES.Web*”，生成项目 SloneMES.Server 与 SloneMES.Web 的跨平台版本。执行“*./publish.py SloneMES.Monitor osx.11.0-arm64*”获取编译项目 SloneMES.Monitor 的指令，在项目 SloneMES.Monitor 中打开终端并执行刚才获取的指令，将生成的文件拷贝至 Publish 文件夹。

进入生成项目 SloneMES.Server 跨平台版本的文件夹并打开终端，执行“*dotnet SloneMES.Server.dll*”查看 MES 服务器的提示信息，如图 6-2 所示。接着执行“*dotnet SloneM*

ES.Server.dll run”运行 MES 服务器，如图 6-3 所示。

图 6-2　MES 系统的提示

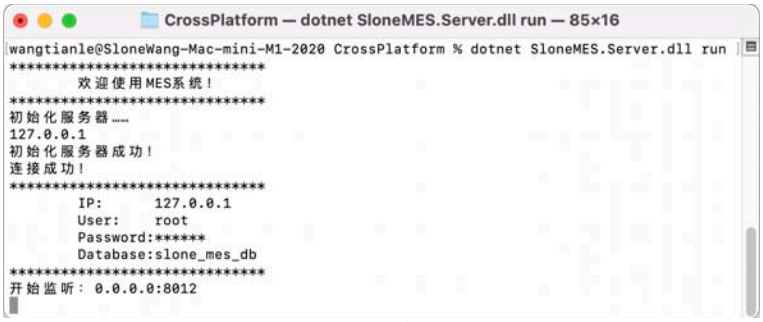


图 6-3　MES 系统的运行

进入生成项目 SloneMES.Web 跨平台版本的文件夹并打开终端，执行“*dotnet Slone MES.Web.dll*”运行 Web 服务器，如图 6-4 所示。接着打开浏览器并访问 https://localhost:5001 进入 Web 页面，如图 6-5 所示。

登录管理员账户 admin 与其他客户用户 customer 后的页面如图 6-6 与 6-7 所示。

经测试，Web 页面中的各项功能均可正常使用。于此同时，MES 服务器的命令行界面与 Web 服务器的命令行界面会显示互相收发的信息，并且使经过加密的，如图 6-8 所示，说明加密模块也是可以正常使用的。

运行 macOS-ARM64 版本的现场监控大屏程序 SloneMES.Monitor.app，其界面正常显示，并且可以连接到 MES 服务器，如图 6-9 所示。

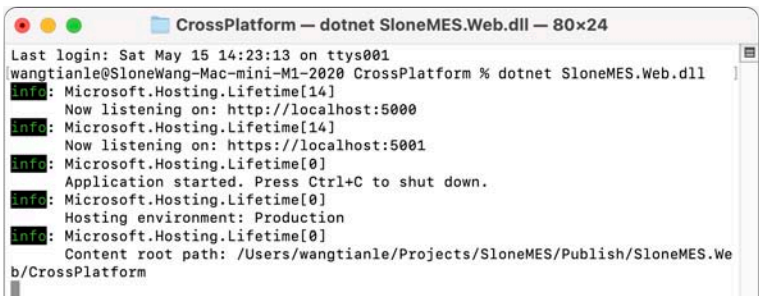


图 6-4　Web 服务器的运行

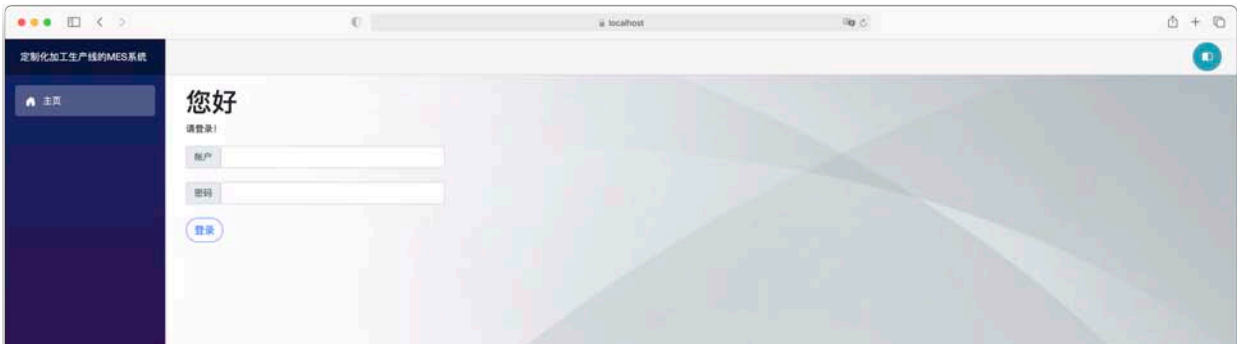


图 6-5　Web 页面的运行

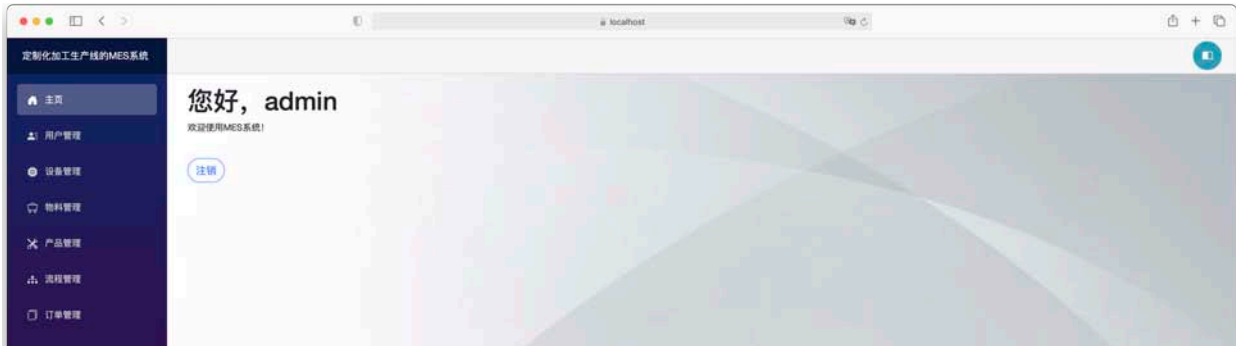


图 6-6　Web 页面（登录 admin 账户）

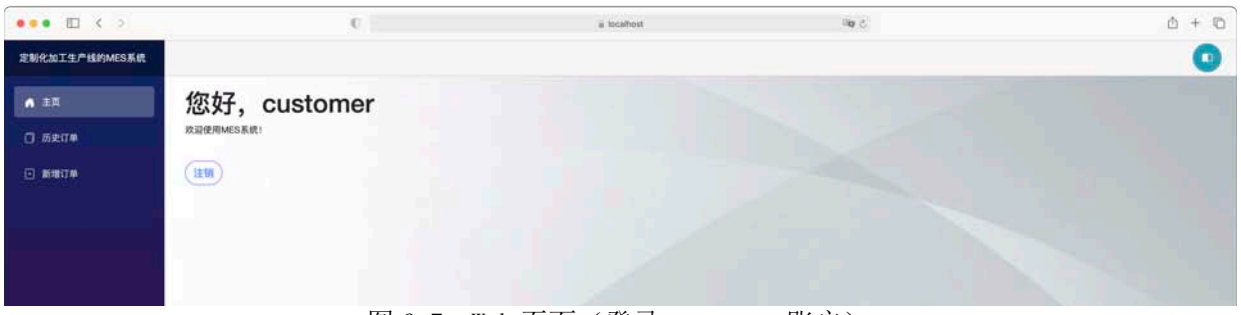


图 6-7　Web 页面（登录 customer 账户）

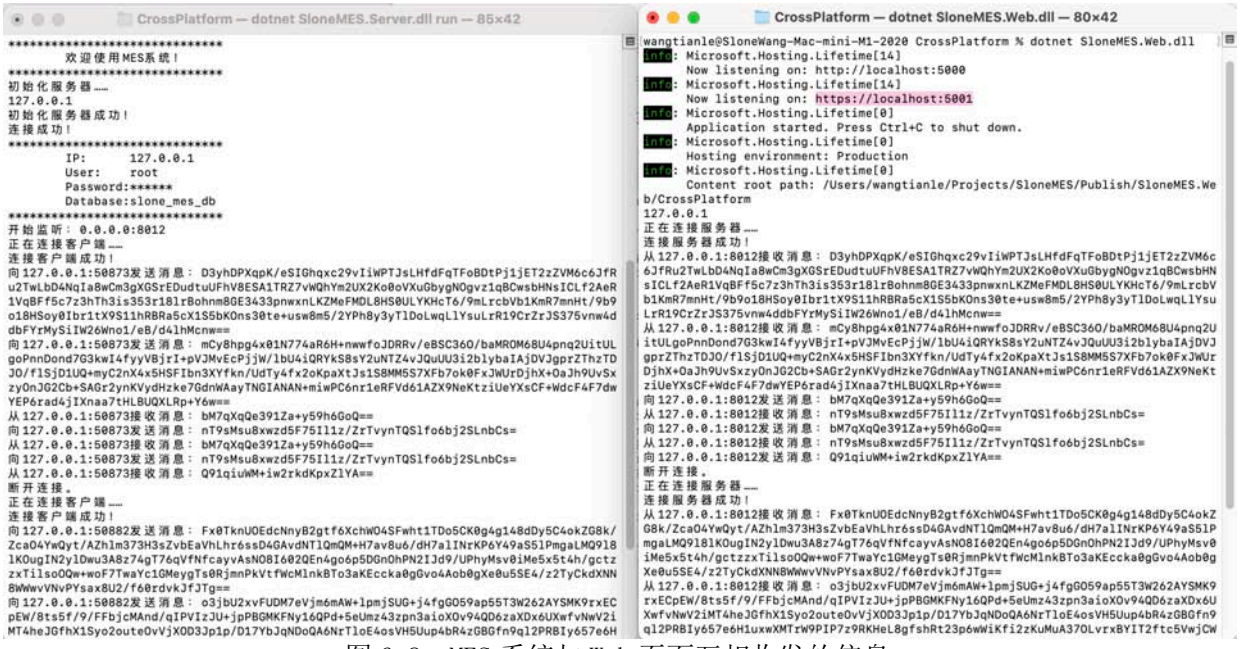


图 6-8　MES 系统与 Web 页面互相收发的信息

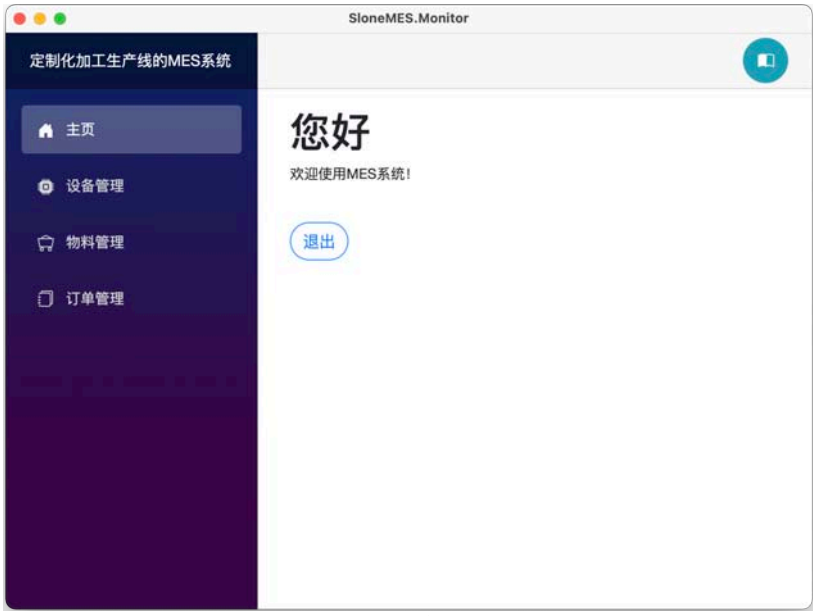


图 6-9　现场监控大屏程序的运行

6.2.2　生产现场调试

将生产线上电，并通过生产线各个站配备的HMI 将每个站都设置为 MES 模式，如图 6-10 所示。

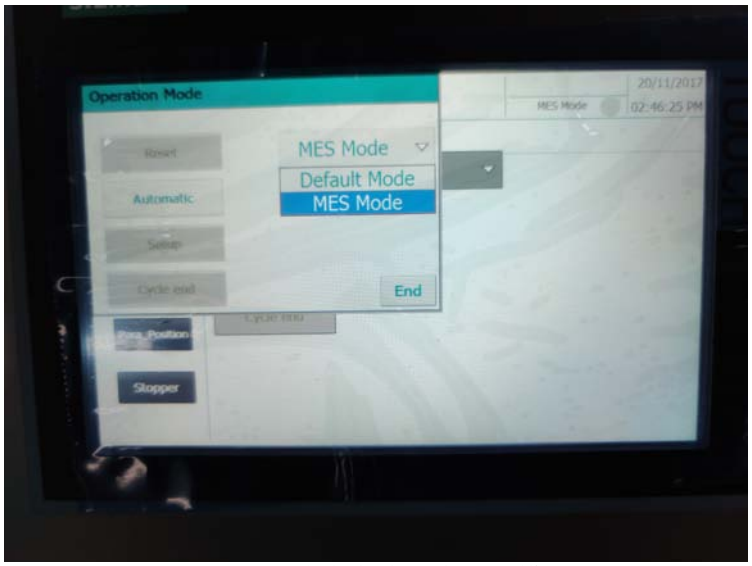


图 6-10　将生产线设为 MES 模式

通过网线连接将电脑以及各单元的 PLC 连接到供料单元的西门子 S615 三层网络交换机，将电脑的 IP 地址设为 172.21.0.90，4 个单元的 PLC 的 IP 地址设为 172.21.1.1、172.21.2.1、172.21.3.1、172.21.4.1，运行 MES 服务器并打开 Web 页面与现场监控大屏程序，Web 页面登录 admin 账户。

进入 Web 页面中的设备管理页面，将各个单元的 PLC 添加进 MES 系统。接着，在物料管理页面添加生产线上所使用的原料。然后，在产品管理页面中添加该生产线可以生产的产品的模板，如图 6-11 所示。图中的弹出框为编号 P#0001“红瓶白盖”产品所用的

物料，数量为 0 的物料将在下订单时根据需要设定数量，提现了生产的定制化。

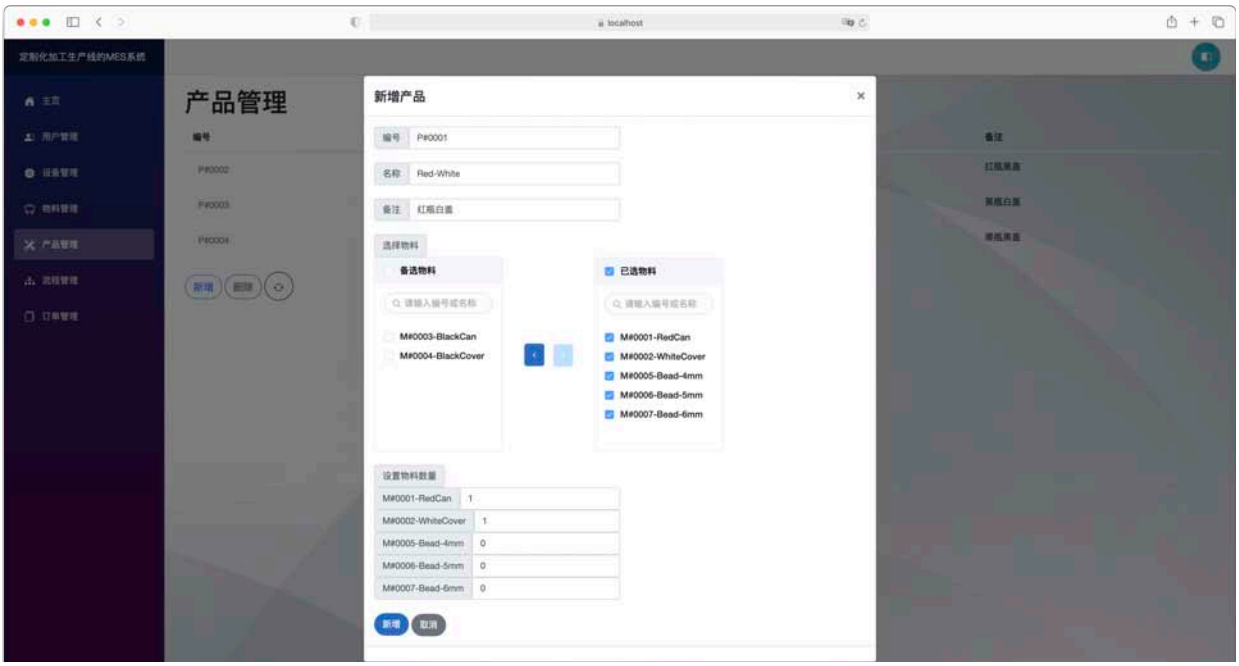


图 6-11　添加产品模板到 MES 系统

准备工作做完后，进入订单管理页面创建订单，如图 6-12 所示。该订单生产了 10 个“黑瓶白盖”的产品，并且每个瓶里含有 3 个 Φ4 圆珠、8 个 Φ5 圆珠和 6 个 Φ6 圆珠。订单创建完成后，MES 服务器会将该订单加入待生产的订单队列，并根据当前设备状况自动安排订单的生产。如果订单不是由管理员创建的，则需要管理员批准后才会加入该订单队列，如图 6-13 所示。除管理员外的其他用户只能看到自己创建的订单，如图 6-14 所示。在生产线空闲时，MES 服务器下达订单执行的指令，生产线制作产品如图 6-15 所示。该图中正在制作的产品即为 O#0001 订单中所定制的产品。当前正在罐装单元进行罐装圆珠。

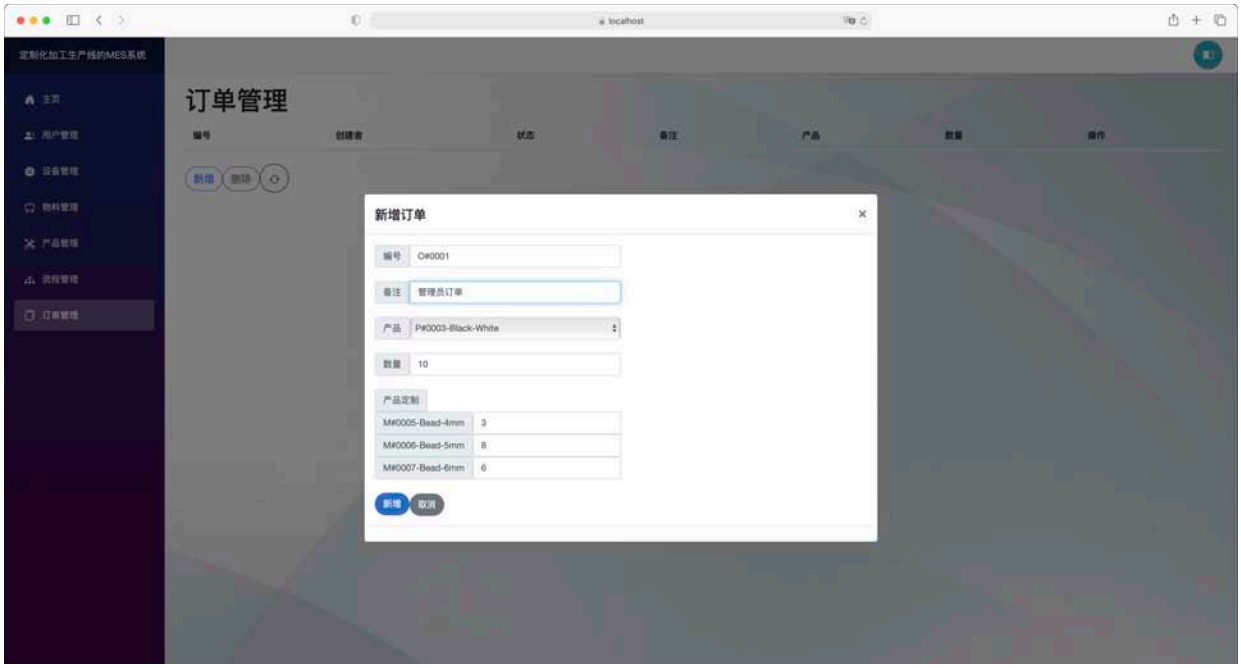


图 6-12　管理员创建订单

图 6-13　管理员查看订单

图 6-14　非管理员查看订单

图 6-15　生产线在 MES 系统控制下运行

在 Web 页面中的设备管理页面中可以查看设备的状态统计和能耗统计图表，如图 6-16、图 6-17 所示，在物料管理页面中可以查看物料使用统计图表，如图 6-18、图 6-19 所示。

在现场监控大屏程序中可以实时查看 PLC 以及订单的状态和物料的剩余情况，如图 6-20、6-21、6-22 所示。

在 Windows 上运行现场监控大屏程序时可能会碰到程序已经在后台运行但是无窗口显示的问题。经过网上查找相关资料，发现是某些电脑上的 Winsock 协议配置有问题，使用管理员身份执行“netsh winsock reset”重置 Winsock 并重启电脑即可解决此类问题。

经测试，在 Web 页面上下订单，订单可以被发送的 MES 服务器上，并被生产线执行，也可在现场监控大屏上看到订单以及PLC 的状态；本课题中的所有程序在Windows、Linux、macOS 上运行情况完全一致，符合跨平台的设计要求。综上所述，MES 服务器与 Web 页面、现场监控大屏程序的调试通过，MES 系统的各项功能均完成课题设计的要求。

图 6-16　设备总览及设备运行状态统计

图 6-17　设备耗电情况统计

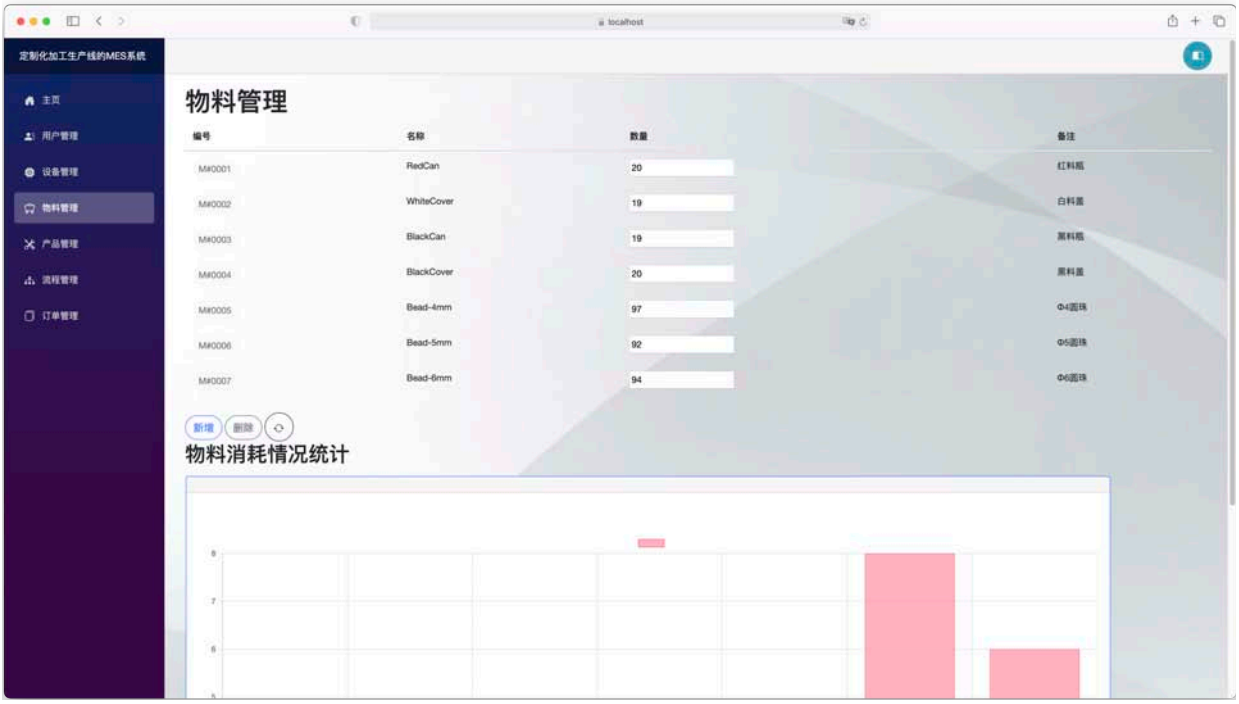


图 6-18　物料总览

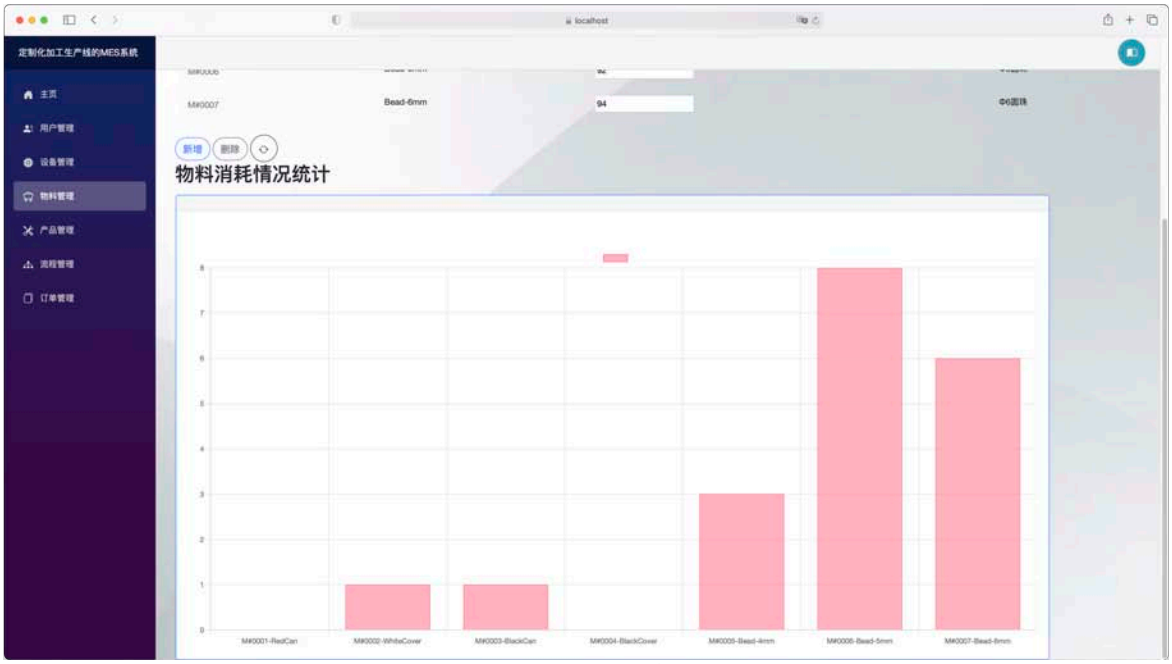


图 6-19　物料消耗情况统计

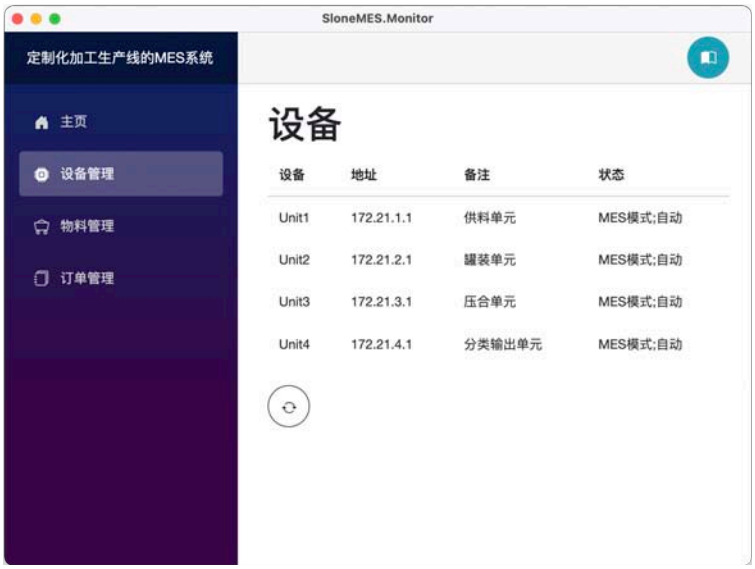


图 6-20　现场监控大屏程序中 PLC 状态

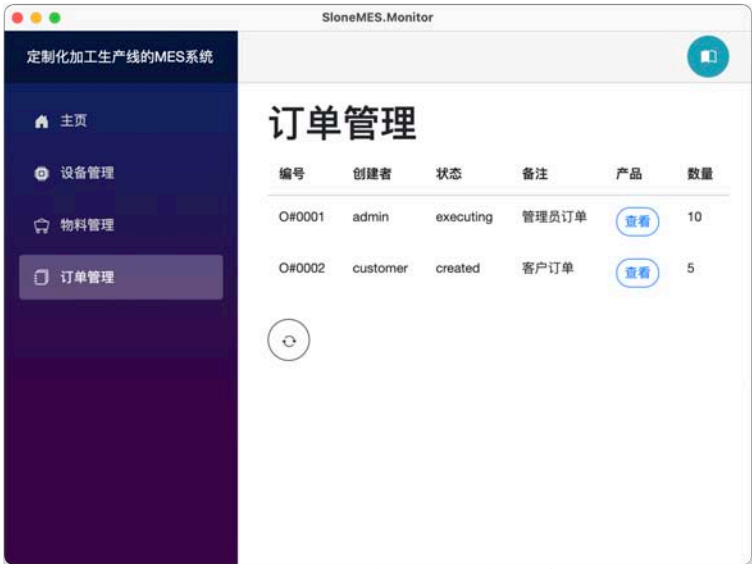


图 6-21　现场监控大屏程序中订单状态

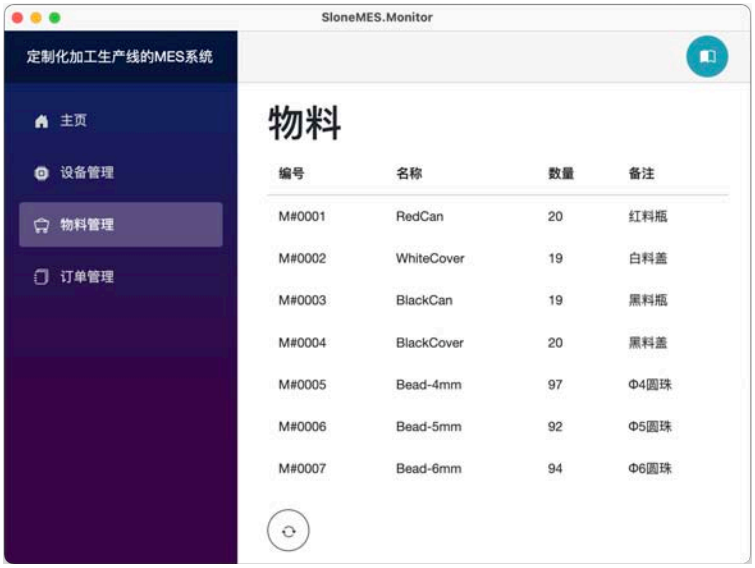


图 6-22　现场监控大屏程序中物料剩余情况

6.3 总结

本课题研究了定制化加工生产线的 MES 系统设计，完成了以下工作：

① 分析了定制化加工生产线模块化组合、标准化架构，进行控制逻辑关系分析，提出了 MES 系统的控制需求；

② 收集了 MES 系统设计与开发的相关资料，比较了各个 MES 系统解决方案的优缺点，提出了整体方案；

③ 设计了 MES 系统的整体架构，完成了智能物料分拣系统的生产流程图、MES 软件部署图、用户界面设计图等设计；

④ 编写了 MES 系统的程序，实现了 MES 系统各个模块的功能，完成了 MES 系统各个模块的调试；

⑤ 将程序发布到服务器和移动终端，完成了定制化加工生产线在 MES 系统控制下的现场调试和试运行。

通过去查阅相关资料，对 MES 系统有了一个初步的认识，并产生了深厚的兴趣。通过几个个月的学习和实践，基本完成了 MES 系统的设计与调试，并取得了预期的结果。

目前，随着工业 4.0 和智能制造 2025 的提出和推进，MES 系统正快速发展。MES 系统是工厂、设备、客户之间的桥梁，使用了 MES 系统的定制化加工生产线具备了以下特点：

① 信息和指令的传达速度得到提升；

② 订单及原材料管理更加便捷；

③ 生产效率更高、系统稳定性更好；

④ 产品可追踪可追溯，生产活动可见可控。

参考文献

[1] 百度. MES（制造企业生产过程执行管理系统）_百度百科[EB/OL]. baike.baidu.com/item/MES/12006591, 2021-01-25/2021-04-06.

[2] Wikipedia. 制造执行系统 - 维基百科，自由的百科全书[EB/OL]. zh.wikipedia.org/wiki/制造执行系统, 2021-01-12/2021-04-06.

[3] ANSI/ISA 95.00.01-2010. Enterprise-Control System Integration - Part 1: Models And Terminology[S].

[4] MESA. MESA International - MESA Model[EB/OL]. www.mesa.org/en/modelstrategicinitiatives/MESAModel.asp, 2021-04-06.

[5] MES centrum. MES - Manufacturing Execution System[EB/OL]. mescenter.org/en/articles/108-mes-manufacturing-execution-system, 2021-04-06.

[6] 胡锦涛. 高举中国特色社会主义伟大旗帜 为夺取全面建设小康社会新胜利而奋斗 ——在中国共产党第十七次全国代表大会上的报告[R]. 北京:人民出版社,2007.

[7] 习近平. 决胜全面建成小康社会 夺取新时代中国特色社会主义伟大胜利 ——在中国共产党第十九次全国代表大会上的报告[R]. 北京:人民出版社,2017.

[8] 百度. 制造执行系统_百度百科[EB/OL]. baike.baidu.com/item/制造执行系统/6039713, 2021-01-25/2021-04-06.

[9] 乔运华,何山,王鹏,李文博,刘晨燕. 车间制造执行系统（MES）在电子装配行业大批量批次生产模式下的应用研究——以 J 公司为例[J]. 制造业自动化,2020,42(11):71-74.

[10] 瞿梦菊.半导体封装行业 MES 系统的设计与研究[J]. 电子测试,2020,(23):139-140.

[11] DB-Engines. DB-Engines Ranking - popularity ranking of database management systems[DB/OL]. db-engines.com/en/ranking, 2021-04-06.

[12] 孙泽君,刘华贞. MySQL 8 DBA 基础教程[M]. 北京:清华大学出版社,2020-06.

[13] 何波, 傅由甲. C#网络程序开发（第二版）[M]. 北京:清华大学出版社,2019-01.

[14] Dino Esposito. Programming ASP.NET Core[M]. Microsoft,2018-09-05.

[15] 彭振云,高毅,唐昭琳. MES 基础与应用[M]. 北京:机械工业出版社,2020-08.

[16] 吴玲. 论基于 eflow 平台的 MES 系统二次开发[J]. 科学与信息化,2018,(016):49-50.

[17] 马朝红. 基于 MES 系统的无纸化技术应用研究[J]. 计算机与网络,2016,(9):55-58.

[18] 宜科(天津)电子有限公司. 信息化解决方案[M/OL]. www.elco-holding.com.cn/attachment/pdf/5ee717258442f84ec19e3877, 2020-06-15/2020-11-30.

[19] 刘诏书,李刚炎. 制造执行系统(MES)标准的综述[J]. 自动化博览,2006,(3):32-35.

[20] SJ/T 11666.9-2016. 制造执行系统（MES）规范 第 9 部分：机械加工行业制造执行系统软件功能[S].

[21] 周春柳. 面向 MES 的加工车间精益生产研究[D]. 大连理工大学,2015.

[22] 禹鑫燚,殷慧武,施甜峰,唐权瑞,柏继华,欧林林. 基于 OPC UA 的工业设备数据采集系统[J]. 计算机科学,2020,47(S2):609-614.

[23] 游俊慧. MVC、MVP、MVVM 三种架构模式的对比[J]. 办公自动化,2020,25(22):11-12+27.

[24] Paolo Ferrari, Alessandra Flammini, Stefano Rinaldi, Emiliano Sisinni, Davide Maffei, Matteo Malara. Impact of Quality of Service on Cloud Based Industrial IoT Applications with OPC UA [J]. Electronics,2018,7(7).

[25] 江凌,杨平利,杨梅,袁媛. 基于 ADO.NET 技术访问 SQL Server 数据库的编程实现[J]. 现代电子技术,2014,37(08):95-98.

附录

1. 设备控制数据中功能编号的说明：

mClass	mNo	功能名称	说明
0x64	0x04	GetFirstOpForRsc	
	0x6F	GetFreeString	
	0x06	GetOpForONoOPos	
0x65	0x01	SetPar	
	0x0A	OpStart	设备开始操作
	0x0F	OpReset	设备急停
	0x14	OpEnd	设备运行结束

2. 创建物料管理数据表、设备管理数据表、产品管理数据表、订单管理数据表的 SQL 语句及其执行结果：

support-files — mysql -u root -p — 86×25

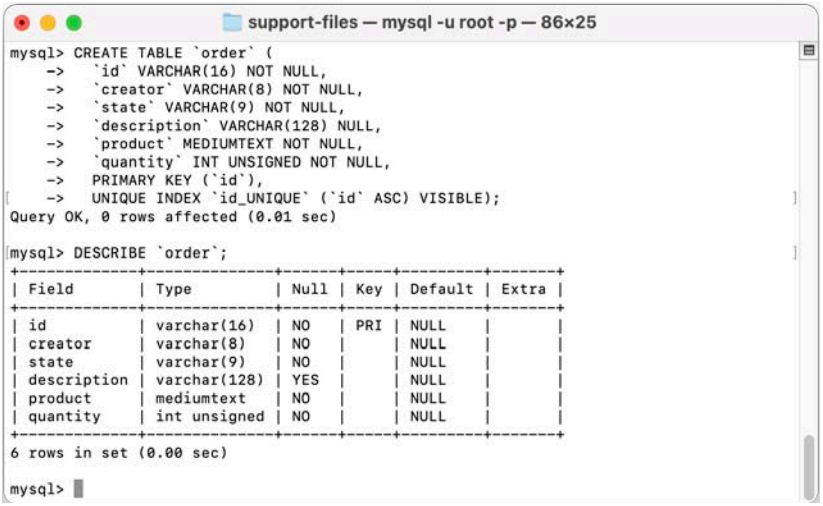
```
mysql> CREATE TABLE `inventory` (
    ->   `id` VARCHAR(16) NOT NULL,
    ->   `name` VARCHAR(32) NOT NULL,
    ->   `quantity_int` INT UNSIGNED NOT NULL,
    ->   `quantity_dec` DECIMAL(10,2) UNSIGNED NOT NULL,
    ->   `description` VARCHAR(128) NULL,
    ->   `isInt` BOOLEAN NOT NULL,
    ->   PRIMARY KEY (`id`),
    ->   UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE);
Query OK, 0 rows affected, 1 warning (0.01 sec)

mysql> DESCRIBE `inventory`;
+--------------+-------------------------+------+-----+---------+-------+
| Field        | Type                    | Null | Key | Default | Extra |
+--------------+-------------------------+------+-----+---------+-------+
| id           | varchar(16)             | NO   | PRI | NULL    |       |
| name         | varchar(32)             | NO   |     | NULL    |       |
| quantity_int | int unsigned            | NO   |     | NULL    |       |
| quantity_dec | decimal(10,2) unsigned  | NO   |     | NULL    |       |
| description  | varchar(128)            | YES  |     | NULL    |       |
| isInt        | tinyint(1)              | NO   |     | NULL    |       |
+--------------+-------------------------+------+-----+---------+-------+
6 rows in set (0.00 sec)

mysql>
```

support-files — mysql -u root -p — 86×19

```
mysql> CREATE TABLE `plc` (
    ->   `name` VARCHAR(32) NOT NULL,
    ->   `IPv4` VARCHAR(15) NOT NULL,
    ->   `description` VARCHAR(128) NULL,
    ->   PRIMARY KEY (`IPv4`),
    ->   UNIQUE INDEX `IPAddress_UNIQUE` (`IPv4` ASC) VISIBLE);
Query OK, 0 rows affected (0.01 sec)

mysql> DESCRIBE `plc`;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| name        | varchar(32)  | NO   |     | NULL    |       |
| IPv4        | varchar(15)  | NO   | PRI | NULL    |       |
| description | varchar(128) | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql>
```

support-files — mysql -u root -p — 86×21

```
mysql> CREATE TABLE `product` (
    ->   `id` VARCHAR(16) NOT NULL,
    ->   `name` VARCHAR(32) NOT NULL,
    ->   `materials` MEDIUMTEXT NOT NULL,
    ->   `description` VARCHAR(128) NULL,
    ->   PRIMARY KEY (`id`),
    ->   UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE);
Query OK, 0 rows affected (0.01 sec)

mysql> DESCRIBE `product`;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| id          | varchar(16)  | NO   | PRI | NULL    |       |
| name        | varchar(32)  | NO   |     | NULL    |       |
| materials   | mediumtext   | NO   |     | NULL    |       |
| description | varchar(128) | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql>
```



3. User 类中的属性、构造函数、解构函数、枚举和静态类：

```
public string Username { get; }
public string Password { get; }
public Roles Role { get; }
public string Description { get; }
public User(in string username, in string password, in Roles role, in string description) =>
    (this.Username, this.Password, this.Role, this.Description, this.Error, this.IsFailed) = (username, password, role, description, "", false);
public void Deconstruct(out string username, out string password, out Roles role, out string description) =>
    (username, password, role, description) = (this.Username, this.Password, this.Role, this.Description);
public enum Roles
{
    /// <summary>
    /// 管理员
    /// </summary>
    admin,

    /// <summary>
    /// 其他
    /// </summary>
    other,

    /// <summary>
    /// 无
    /// </summary>
    none,
}
public static class RolesStr
{
    /// <summary>
    /// 管理员
    /// </summary>
    public const string admin = "admin";

    /// <summary>
    /// 其他
    /// </summary>
    public const string other = "other";

    /// <summary>
    /// 无
    /// </summary>
    public const string none = "";
}
```

4. Order 类的订单状态枚举：

```
public enum OrderState
{
    /// <summary>
    /// 创建
    /// </summary>
    created,

    /// <summary>
    /// 排队
    /// </summary>
    queuing,

    /// <summary>
    /// 执行
    /// </summary>
    executing,

    /// <summary>
    /// 结束
    /// </summary>
    finished,

    /// <summary>
    /// 中断
    /// </summary>
    interrupt,

    /// <summary>
    /// 失败
    /// </summary>
    failed,

    /// <summary>
    /// 删除
    /// </summary>
    deleted,

    /// <summary>
    /// 空
    /// </summary>
    none,
}
```

致谢

光阴似箭，日月如梭大学四年的学习生涯已然接近尾声。在这四年的生活和学习中，有许许多多的老师和同学在我遇到困难时都给予了我非常大的帮助，在此我想对他们表示深深的感谢。

从一开始课题的选择，到后来项目的实施，再到最后论文的攥写，都离不开史艳霞教授的悉心关怀和谆谆教诲，十分感谢史艳霞教授在各个环节给予的细心指导，史老师严谨治学的态度、丰富渊博的知识以及诲人不倦的师者风范是我终生学习的楷模。特别感谢天津科技大学的刘振全教授，刘老师多次审阅论文，对细节进行修改，为本文的撰写提供了许多宝贵的意见。还要感谢企业的各位老师和工程师对本课题提供硬件上的支持以及技术上的帮助。感谢我组各位组员对本课题中生产线的硬件以及 PLC 程序进行讲解与分析。

这四年中还得到众多老师和同学的关心、支持和帮助，在此向他们以及对本文进行审阅、评议和参与答辩的各位老师表示衷心的感谢。