

# 嵌入式实时系统设计模式的研究与应用

**[摘要]**：为满足嵌入式实时系统快速适应硬件型号升级、快速变更要求以及可伸缩、可修改、可复用等方面的需求，本文提出了一种适用于嵌入式实时系统的设计模式——ERTSDP，解决了系统分析设计中实时性、可靠性等方面的问题，提高了软件的开发速度。最后结合具体应用对这一模式的实例化过程加以阐述。

**[关键词]**：设计模式；嵌入式实时系统；远程监控系统；UML

## 1 引言

从系统的观点来看，嵌入式实时系统也是系统，特别是硬件技术的快速发展，嵌入式实时软件也有快速适应硬件型号升级问题，也有业务快速变更要求以及可伸缩、可修改、可复用等方面的问题。但面向对象技术对于实时性、可靠性的描述并不适合，在嵌入式实时应用中依然不是主流设计方法，从而引入了模式的概念。

最早也是最基础的设计模式是Liskov提出的七种基本模式，并提出对象构成模式的五条基本准则<sup>[1-4]</sup>，为面向对象设计模式奠定了理论基础。Gamma等四人提出了常用的25种设计模式<sup>[5]</sup>，为用模式设计软件体系结构提供了工程实践的基础。不过Gamma的模式作用域是局部的，只在单个的协作内。Bruce Douglas将两个重要的软件工程进展——模式和UML相结合，成功地应用在主流实时软件传统使用的概念和技术中<sup>[6]</sup>。

本文讨论的模式ERTSDP是基于Bruce Douglass的实时设计模式基础之上，对典型嵌入式实时系统的设计解决方案加以总结抽象，提出的一套完整的设计思路。这一模式的提出可以促使本领域软件开发速度成倍的提高，更为重要的是软件质量也可以得到保证。

## 2 嵌入式实时系统设计模式(ERTSDP)研究

### 2.1 设计模式的定义

设计模式<sup>[7-9]</sup>是对经常出现问题的泛解。模式由三个重要的方面组成：第一是问题(problem)，这是要以模式来处理的设计的某个方面的陈述，也就是要用模式解决的某些优化和QoS的侧面。第二是解决方案(solution)，也就是模式本身，模式用指明了角色的结构图表示。最后是结果(consequences)。

### 2.2 ERTSDP (Embedded Real-Time Systems Design Pattern)

#### 1. 目的

针对嵌入式实时系统分析和设计过程中的问题进行抽象并建立模型，使得新系统可以通过模式匹配、实例化等手段快速建立，从而达到缩短嵌入式实时产品开发周期的目的。

#### 2. 解

解即模式本身，2.3小节将详细讨论。

#### 3. 结果

结果是使用模式后的一组利弊，将在本文最后讨论。

### 2.3 模式的解

#### 2.3.1 系统分析

分析的目的在于定义待开发系统的基本性质。一般来说，分析是一个黑箱视图，而设计则按某个服务质量QoS的要求提供充分的功能。对系统的分析又可分为两个阶段：需求分析阶段和系统工程阶段。

##### (1)需求分析阶段

在需求阶段尽可能详细地标识和捕获当前原型的需求，可用顺序图、状态图、活动图、正文描述以及约束等的组合对需求加以描述。

## (2) 系统工程阶段

系统工程阶段实际是做高层的体系结构设计。系统工程阶段主要定义子系统的体系结构、子系统的接口及交互协议、将系统的用例和需求分解为子系统的用例和需求以及对系统的算法分析和控制法则规范说明。

### 2.3.2 系统设计

设计就是该问题的具体解决方案，是对分析模型的优化。优化准则的集合就是要求的系统服务质量 QoS。如果把每个 QoS 方面看作是独立的特征并具有相对重要性的加权因子，那么一个好的设计就是找出下面公式的最小值。

$$\text{Min} [ \text{QoSFeature}_j * \text{Weight}_j ]$$

其中  $\text{Weight}_j$  指的是与第  $j$  个  $\text{QoSFeature}$  相关的相对重要性。对系统的设计也分为两个阶段：体系结构设计阶段和详细设计阶段。

#### (1) 体系结构设计阶段

体系结构包括逻辑体系结构和物理体系结构。逻辑体系结构只涉及模型本身如何组织，这种组织可简可繁，取决于小组需要用它构造什么；物理体系结构指的是组织存在于运行时的事物。这一阶段用五层体系结构视图、子系统视图和资源并发视图等来描述。

**五层体系结构视图**<sup>[6]</sup>是一个特定的体系结构，作用于许多嵌入式和实时系统的通用结构。它包含五个域：应用域(application domain)、用户界面域(user interface domain)、通信域(communication domain)、抽象操作系统域(abstract OS domain)和抽象硬件域(abstract hardware domain)。

**子系统视图**是表示重要子系统的类图，多用于对系统进行细化时。

**资源并发视图**是实时和嵌入式系统一个最突出的特点。一个有资源服务的元素，它的有效性由一个或多个服务质量(QoS)特性来定义。QoS 是资源的量化性质，如容量、执行速度、可靠性等等。

#### (2) 详细设计阶段

详细设计阶段是对对象和类的内部精细加工，它只限制在单个对象和类之中。详细设计时，多数优化都集中在数据构造、算法分解、对象状态机的优化、对象实现策略、关联的实现以及可见性和封装问题等方面。

模式是对设计中一般问题的抽象，所以针对具体应用的模式匹配以及实例化是可根据需要对 ERTSDP 变动的。譬如，需求分析阶段可以根据问题的复杂性对描述手段任意组合；系统工程阶段也是微周期中的可选部分；如果系统非常复杂，可以递归调用子系统视图对系统分解、分级细化；还可在体系结构设计时增加 Activity 图实现对任务的划分，增加 sequence 图对系统运行流程描述。

总之，在具体应用中可以灵活的应用模式来辅助设计。下面通过对常见的嵌入式实时系统——远程监控系统应用 ERTSDP 模式进行分析和设计，阐述了实例化模式的方法，同时也是对这一模式的讨论和验证。

## 3 基于 ERTSDP 的远程监控系统

### 3.1 系统需求分析

远程监控系统现在已经深透到社会的各个方面，简而言之，远程监控系统就是将现场设备的运行数据发送至远端加以监测和控制。本系统投入运行后预期的用户有三类：普通用户、系统管理员、故障专家。他们具有不同的职责，因而赋予了不同的权限。图 1 的 Use Case 框图可以清晰地反映出他们各自的权限职责。

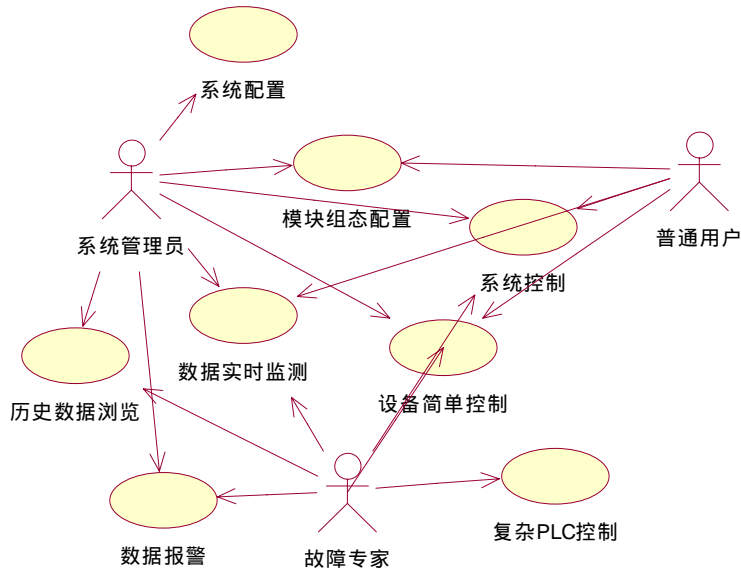


图 1 远程监控系统的 Use Case 框图

### 3.2 系统体系结构设计

本系统包含三个子系统——NetWeb、NetIO 和 RemoteClient(见图 2)。NetWeb 是整个系统的核心,承担着实时/历史数据管理、报警条件检测,存贮用户组态信息等功能;NetIO 用于实现现场数据的采集(输入)和装置的控制(输出);RemoteClient 则实现了客户端图形用户界面的功能。其中 NetWeb 和 NetIO 通过 RS485 总线进行通讯,NetWeb 和 RemoteClient 则通过 socket 进行信息交换。

NetWeb 运行在嵌入式开发板上,操作系统是自主研发的嵌入式实时 Linux,采用的方案设计是 uClinux+RTAI。

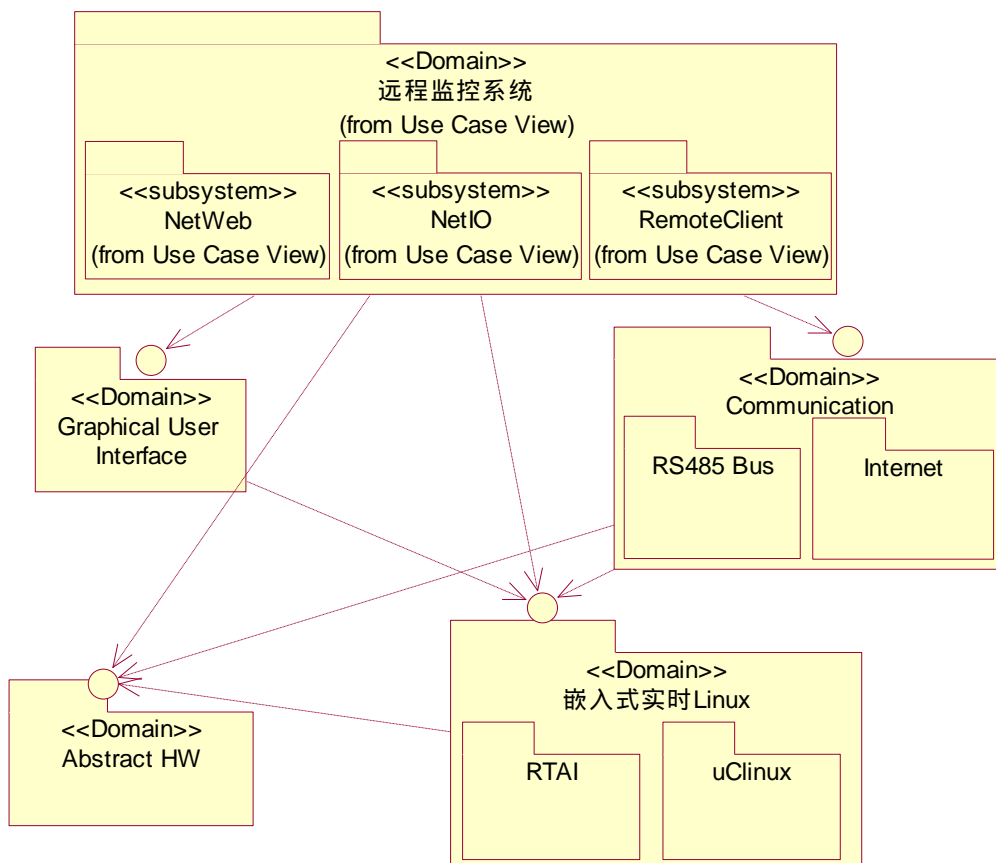


图 2 远程监控系统的五层体系结构视图

### 3.3 子系统视图

子系统视图是表示重要子系统的类图，多用于对系统进行细化时。图 3 是细化后的 NetWeb 子系统，从图中可以看到，特殊的双内核 OS——uClinux+RTAI 决定了 NetWeb 模块分为实时应用和非实时应用。下面将从初始化和运行两方面对图 3 详细解释。

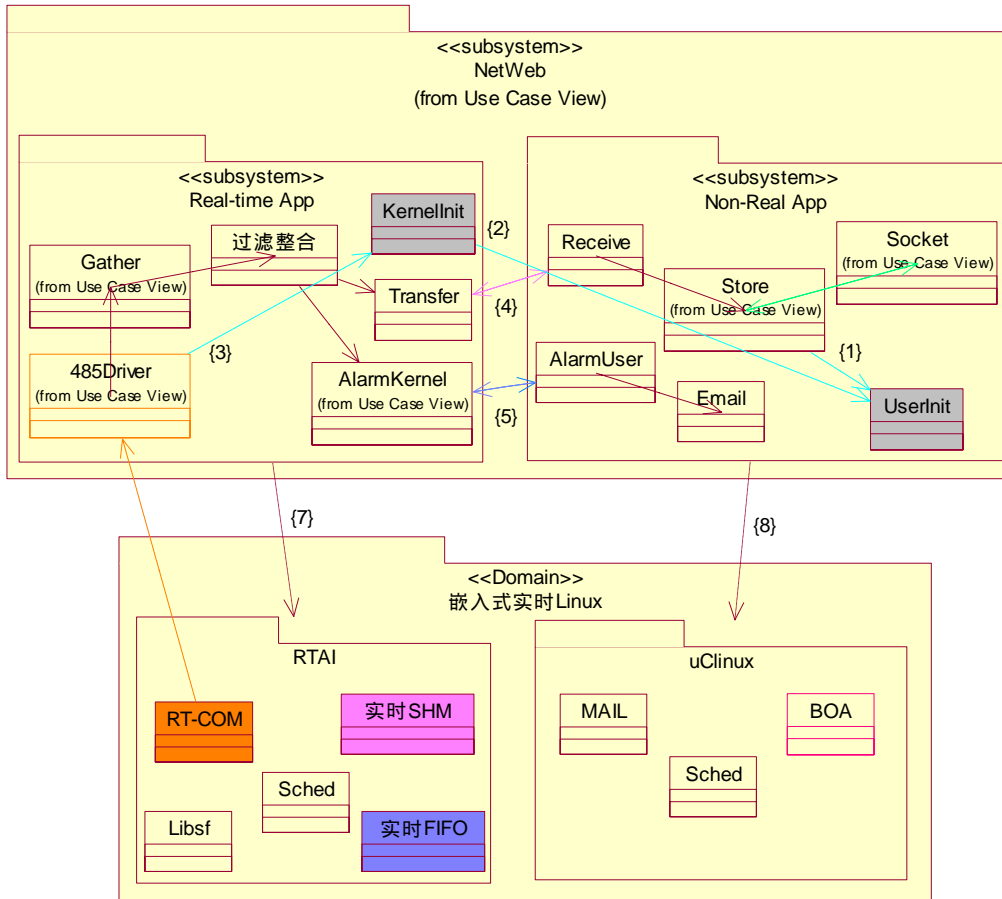


图 3 NetWeb 模块的子系统视图

### 3.3.1 NetWeb 子系统初始化

NetWeb 子系统初始化过程主要由 KernelInit 模块和 UserInit 模块(图中灰色的模块)实现。首先 KernelInit 模块执行，完成内核空间的初始化任务；然后启动 UserInit 模块，这个过程比较复杂，首先 Store 模块将系统存储的配置信息和数据信息传给 UserInit 模块(标注{1}所示)，然后 KernelInit 模块驱动 485Driver 模块实时采集当前模块的信息，传送给 UserInit 模块(标注{3}、{2}所示)，UserInit 模块经过两者信息的比较，确定采集的对象，以及模块信息的变更、系统状态的变更等，最终完成初始化工作。

### 3.3.2 NetWeb 子系统运行流程

当系统完成初始化开始运行后，Gather 模块调用 485Driver 模块相关接口 API 对底层 32 个 NetIO 模块进行轮询式采集，采集上来的数据经过过滤整合后，正常的数​​据传​​送至 Transfer 模块，这时调用 RTAI 内核中的 RT\_SHM 模块实现内核和用户空间大规模数据的共享(见标注{4})。Receive 模块从 RT\_SHM 接受到数据，送到 Store 模块存储，然后根据用户要求送到 Socket 模块传到远程客户端。如果过滤整合时发现数据异常，就驱动 AlarmKernel 模块，这个模块启动 RTAI 的 RT\_FIFO 模块，将需报警数据通过管道(见标注{5})传​​送至用户空间的 AlarmUser 模块。AlarmUser 模块再驱动 Email 模块以邮件的形式完成报警。

## 3.4 资源并发视图

本系统中任务间资源共享的方式主要有两种：共享内存和消息队列。图 4 和图 5 分别对这两种资源共享方式采用并发视图进行了分析。

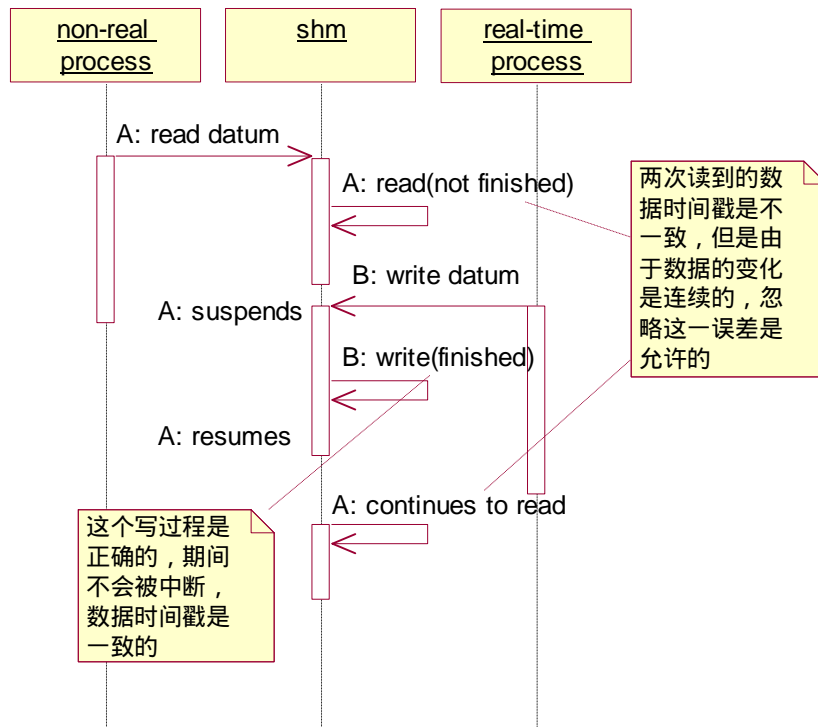


图 4 共享内存访问并发视图

共享内存对于大规模的数据共享非常适用，这里讨论的是 RTAI 提供的实时 SHM。RT\_SHM 并没有提供任何互斥机制来控制实时进程和非实时进程访问同步性(对于实时任务或非实时任务之间的共享有保护措施，故不再考虑)，虽然可以自己实现一些 lock 机制来控制，但是考虑到内核任务的优先级远高于用户任务，即实时任务从来不会被非实时任务中断，所以实时任务对共享区写数据时永远是一次性写完所有模块信息，而且这些数据的时间戳是一致的，可以保证数据的正确性(如图 4)；而非实时任务读共享数据时是可能发生过程中被中断的情况，导致可能一次读取的模块数据的时间戳不一致。但实际中写进程的速度远远大于读进程的速度，同时考虑到实际运行时模块数据变化是连续的，同一模块不同时间戳采集的数据变化率不会很大，而现阶段读进程只要满足用户需要的刷新频率(QoS)就足够了，所以对于数据部分丢失现阶段并不可惜。

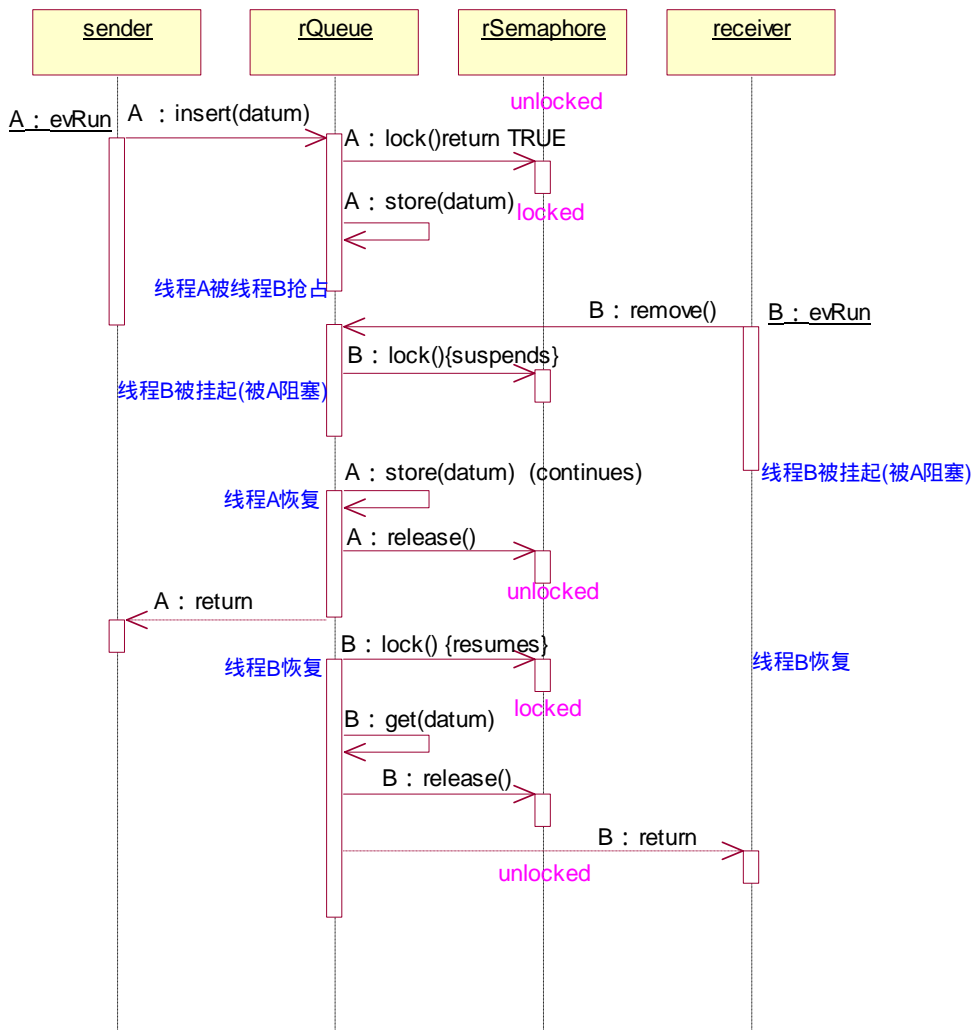


图 5 消息队列访问并发视图

消息队列提供了一种传输多条消息的机制。如图 5 所示，Sender 线程因响应 evRun 事件而运行。Sender 类调用 rQueue::insert()操作，并锁定信号灯。在 rQueue::insert()操作过程中，该线程被更高优先级的 Receiver 线程中断。Receiver 试图检查在其队列中新加入的消息。它调用 rQueue::remove()，同时试图锁定信号灯。如果失败，Receiver 自动挂起或被阻塞，因为信号灯已经被锁定到 Sender 线程，这使得 Sender 线程继续工作，完成 store()操作。一旦完成，rQueue 类释放它的信号灯。它将解除对 Receiver 线程的阻塞，Receiver 线程这时可以锁定信号灯，读取等待在队列中的消息，解除信号灯的锁定，处理收到的消息。

#### 4 系统仿真平台

通过应用 ERTSDP 模式对远程监控系统分析、设计以及实现，满足了用户预期的需求。为了对系统功能、性能进一步验证，搭建了系统仿真实验平台(如图 6)。图中标注为 NetWeb 的模块就是系统的嵌入式装置，这一模块内部是 net-start 开发板，操作系统是嵌入式实时 Linux，子系统 NetWeb 的所有功能就运行其上。并排的五模块都是 NetIO 模块，分别连接有不同的输入/输出信号，可以看到它们是通过 RS - 485 与 NetWeb 相连。AI 模块通过传感器采集到灯泡(图中黄色圆圈)的温度值，根据温度的高低决定是否启动 DO 模块输出控制风扇的转动。同时 AO 模块可以根据用户需求，经由可控硅的控制可变电压，进而调节灯泡的亮暗程度。

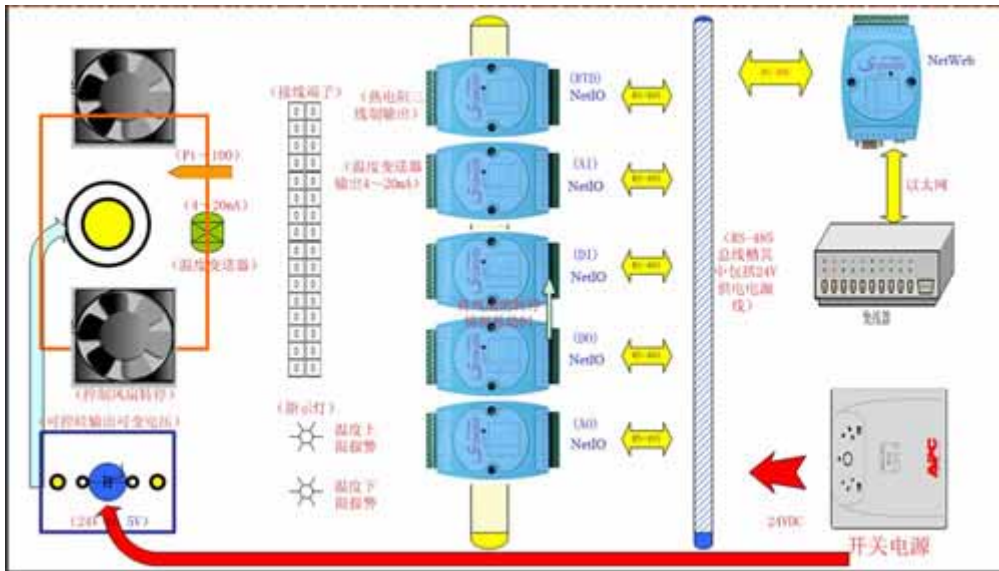


图6 实验系统平台结构

## 5 结论

本文提出的设计模式 ERTSDP 是一种构建系统的完整解决方案，旨在为典型的嵌入式实时应用提供分析设计的模型。实践证明这一模式非常适用于实时性、可靠性的描述，可以有效地提高嵌入式实时领域软件的开发速度，并使软件的质量得以保证。

## References

- [1] Barbara H. Liskov and Jeannette M. Wing, A Behavioral Notion of Subtyping, *ACM Transactions on Programming Languages and Systems*, November 1994.
- [2] Barbara Liskov and Jeannette M. Wing, Family Values: A Behavioral Notion of Subtyping, CMU-CS-93-187 (supersedes CMU-CS-93-149 and CMU-CS-92-220).
- [3] Barbara Liskov and Jeannette M. Wing, Specifications and Their Use in Defining Subtypes, *Proc. of OOPSLA '93*, September 1993.
- [4] Barbara Liskov and Jeannette M. Wing, A New Definition of the Subtype Relation, *Proc. of the European Conference on Object-Oriented Programming '93*, Springer-Verlag LNCS 707, July 1993, pp. 118-141. Also CMU-CS-93-149, April 1993; MIT LCS Programming Methodology Group Memo 76, May 1993.
- [5] Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA: Addison-Wesley, 1995.
- [6] Bruce Power Douglass. *Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems*. Pearson Education, Inc, publishing as Addison Wesley Professional, Copyright 2003.
- [7] Boehm, Barry, Ellis Horowitz, Ray Madachy, Donald Reifer, Bradford Clark, Bert Steece, A. Winsor Brown, Sunita Chualani, and Chris Abts. *Software Cost Estimation with Cocomo II*. Upper Saddle River, NJ: Prentice Hall, January 2000.
- [8] Lea, Doug. *Design Patterns for Avionics Control Systems*, <http://st-www.cs.uiuc.edu/user/patterns/patterns.html>, 1994.
- [9] OOPSLA 2001, Workshop on Patterns in Distributed Real-Time and Embedded Systems, ACM, October 2001.
- [10] Douglass, Bruce Powel. *Doing Hard Time: Developing Real-Time Systems with UML, Objects, Frameworks and Patterns*, Reading, MA: Addison-welsey, 1999.
- [11] D. Schmidt, M. Stal, H. Rohnert, F. Buschmann. *Pattern-Oriented Software Architecture, Volume 2: Patterns for Concurrent and Networked Objects* (Wiley, 2001).

作者：崔珂

Email: ck\_summersun@163.com

