

# 第11章 嵌入式系统的开发设计



# 目 录

- 11.1 嵌入式系统设计方法介绍
- 11.2 基于ARM的嵌入式WEB服务器设计
- 11.3 物联网网关设计实例
- 11.4 智能无人值守实验室监控系统设计实例

# 11.1 Part One

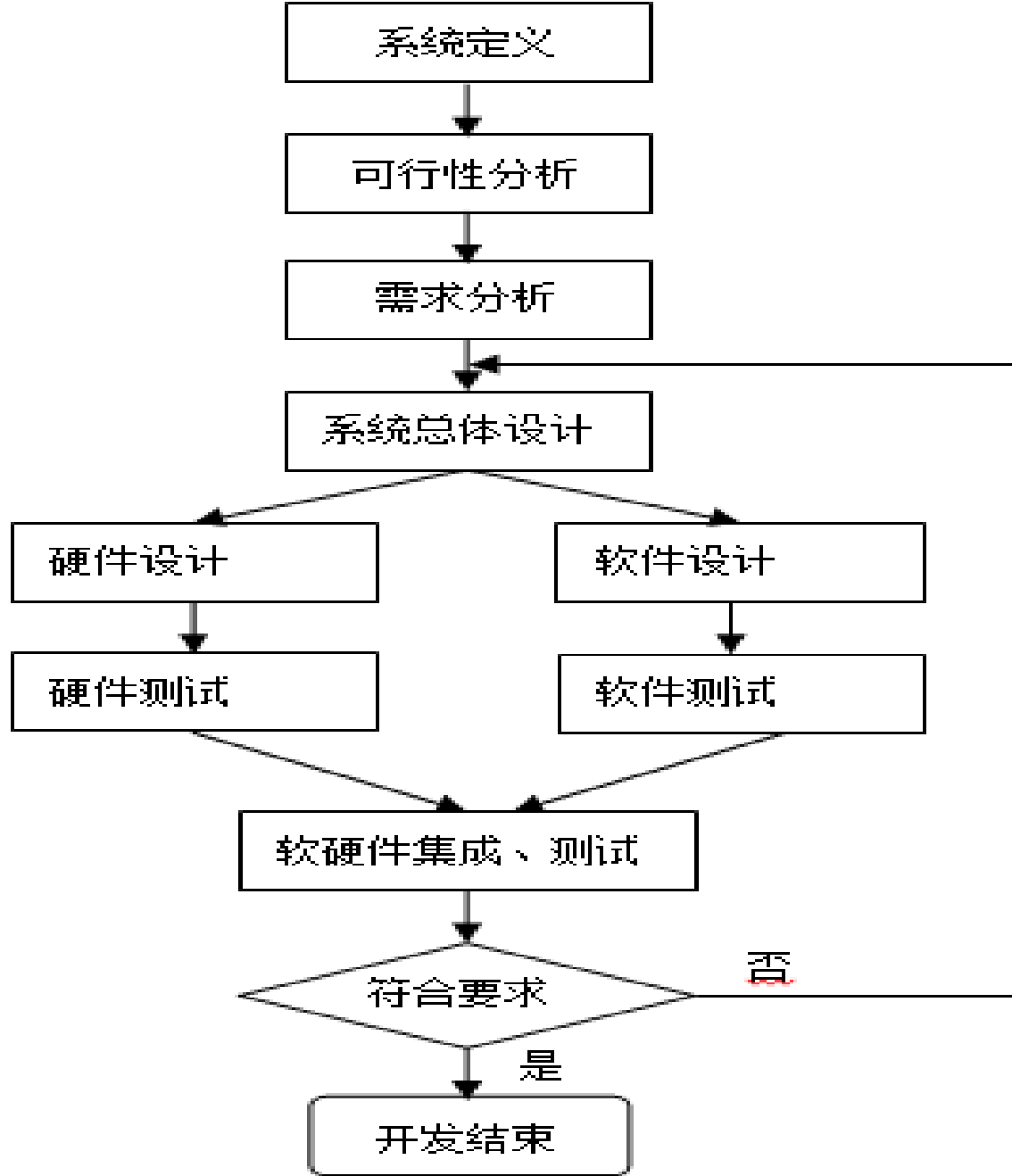
## 嵌入式系统设计方法介绍

---

# 11.1.1 传统的嵌入式系统设计方法

系统定义：系统的设计要求

（注：设计要求一般由用户提出，形成文档。这是项目的来源。设计要求尽可能详细，尽量不要有歧义。）



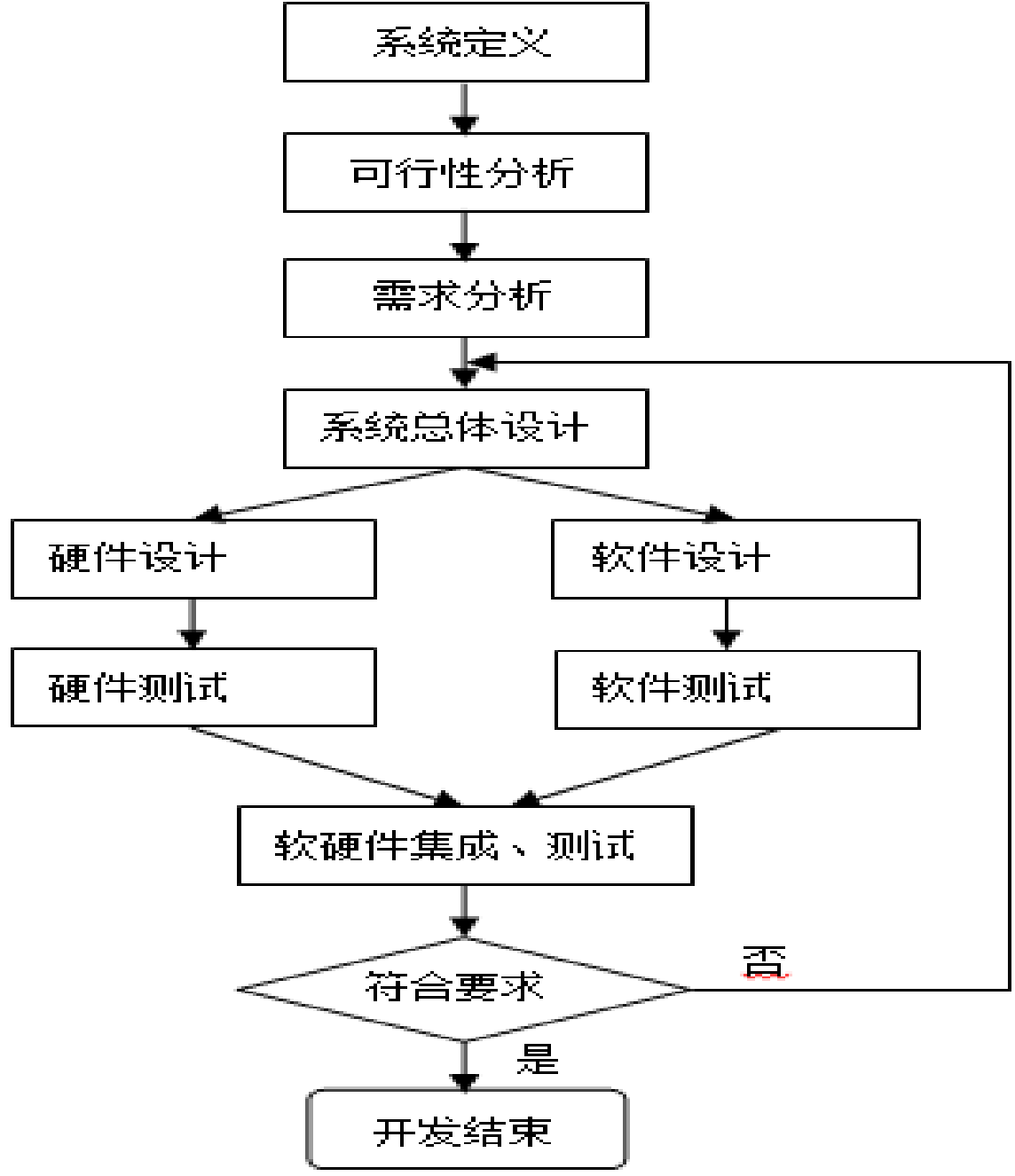
# 便携式网络电视



# 11.1.1 传统的嵌入式系统设计方法

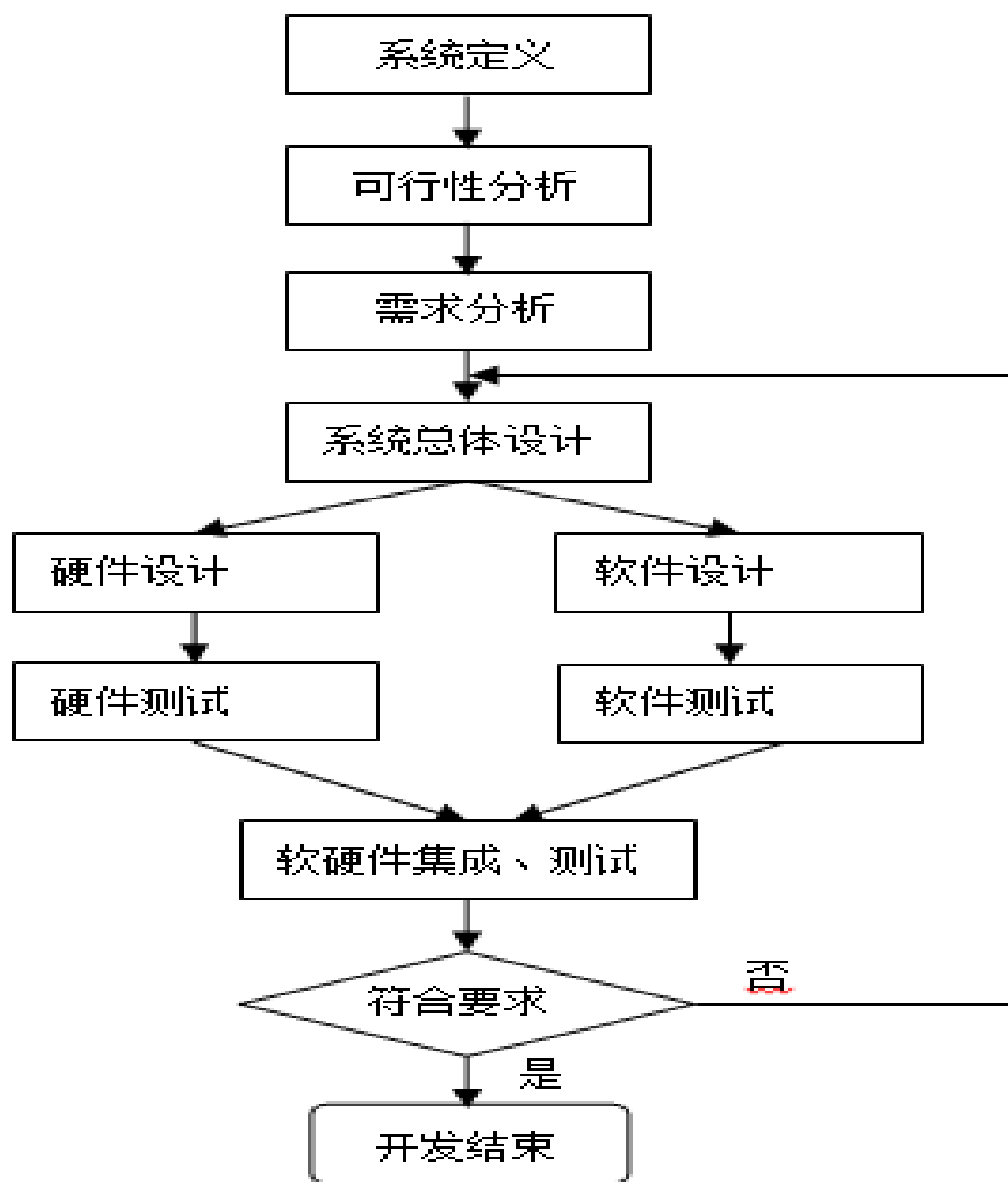
了解客户的业务过程存在哪些需要解决的问题，哪些问题可以通过嵌入式工程项目来解决，哪些不能。

如果不能解决，是否可以通过调整业务流程或业务重组的方式解决。



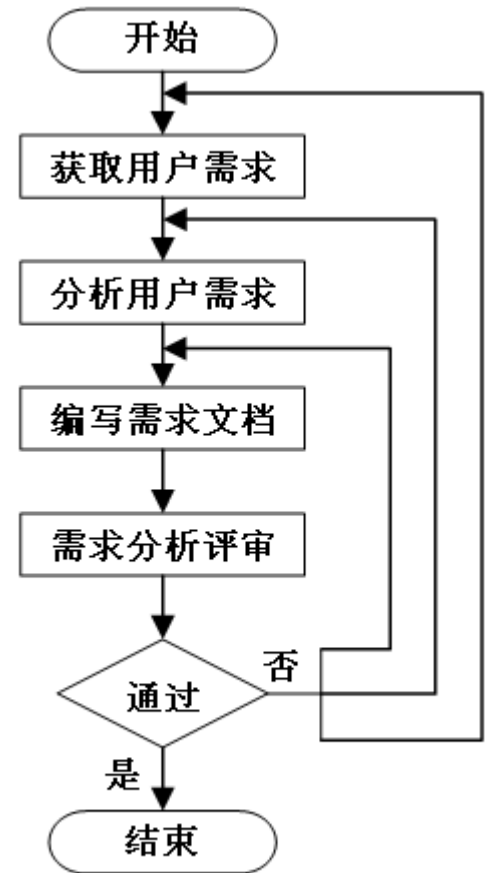
### 11.1.1 传统的嵌入式系统设计方法

在对目标嵌入式系统提出系统定义方案后，要对系统实现进行可行性和需求分析。



# 需求分析阶段

- 需求分析是针对设计要求进行分析，以确定各项要求的可行性，以及相应的实现方法。通过需求分析，可以确定要实现哪种功能、实现到什么程度、技术指标等。
- 需求可分为功能部分和非功能部分。非功能性需求包括了性能、价格、物理尺寸和重量、功耗等方面的因素。
- **建议采用UML建立模型，进行准确描述。**



需求分析的过程



# 需求分析阶段

需求分析表格

名称	
目的	
输入	
输出	
功能	
性能	
生产成本	
功耗	
物理尺寸和重量	

## 便携式网络电视设计的需求分析表格

名称	便携式网络电视
目的	为用户提供移动网络和收看数字电视服务, 同时具有广播和交互式多媒体应用功能
输入	触摸式面板, 电源按钮
输出	LCD显示屏, 内置喇叭
功能	<ul style="list-style-type: none"><li>●电子节目指南</li><li>●高速数据广播</li><li>●软件在线升级</li><li>●因特网接入</li><li>●条件接收</li></ul>
性能	画面流畅清晰, 30fps

# 系统规格说明

- 说明系统做些什么，具有哪些方面的功能
- 是系统开发、验收和管理的依据
- 不能有任何歧义
- 必须认真仔细编写，以便能够精确详尽地反映客户对系统各方面的需求
- 是设计时必须明确遵循的要求和准则

# 需求分析评审

## - 评审的主要内容:

- 系统定义的目标是否与用户的要求一致;
- 系统需求分析阶段提供的文档资料是否齐全;
- 文档中的所有描述是否完整、清晰, 是否准确地反映了用户要求;
- 与所有其它系统的重要接口是否都已经描述;
- 所开发项目的数据流与数据结构是否完整;
- 所有图表是否清楚, 在不补充说明的情况下能否理解;
- 主要功能是否已包括在规定的软件范围之内, 是否都能充分说明;

# 需求分析评审

## - 评审的主要内容：

- 设计的约束条件或限制条件是否符合实际；
- 开发的技术风险是什么；
- 是否考虑过系统需求的其它方案；
- 是否考虑过将来可能会提出的一些要求；
- 是否详细制定了检验标准，这些标准能否对系统定义成功地进行确认；
- 有没有遗漏、重复或不一致的地方；
- 成本进度估算是否受到了影响等。

# 系统总体设计

在经过严格分析论证后，进入到系统总体设计方案阶段。

总体设计是设计的第一步，其目的是描述系统如何实现由系统定义规定的那些功能。它需要解决嵌入式系统的总体构架，从功能实现上对软硬件进行划分；在此基础上，选定处理器和基本接口器件；根据系统的复杂程度确定是否使用操作系统，以及选择哪种操作系统；此外，还需要选择系统的开发环境、软件系统的总体架构设计等。

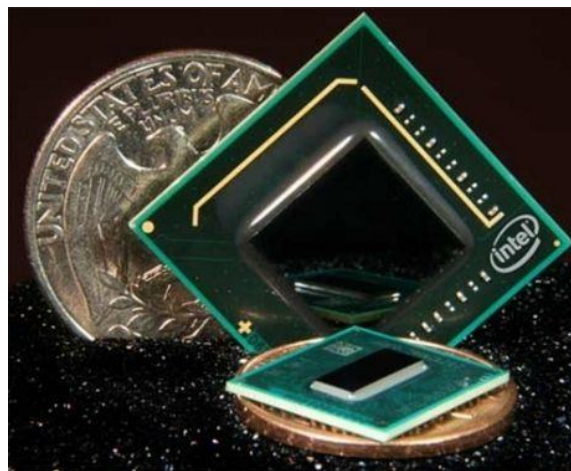
通常硬件和软件的选择包括：

- 处理器
- 硬件部件
- 操作系统
- 编程语言
- 软件开发工具
- 硬件调试工具
- 软件组件等。

# 处理器的选择



Cortex A8, ARM指令集



Intel Atom, X86指令集



龙芯, MIPS指令集



STM32, THUMB指令集



# 处理器的选择

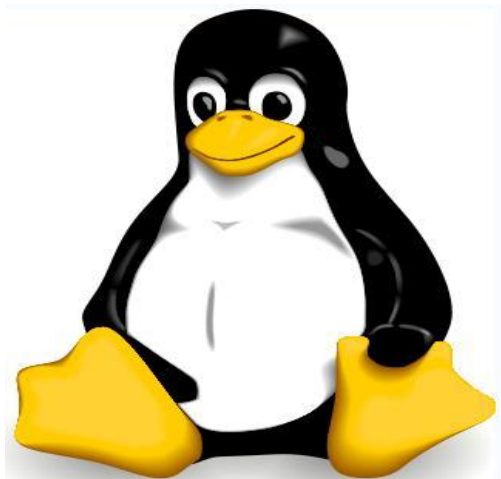
## 处理器选择考虑因素

- 处理器性能
- 处理器技术指标
- 功耗
- 软件支持工具
- 处理器是否内置调试工具
- 供应商是否提供评估板

除此之外，硬件选择要考虑的因素主要还包括：

- 生产规模
- 开发的市场目标
- 软件对硬件的依赖性
- 最后只要可能，尽量选择使用普通的硬件。

# 操作系统的选择



AliOS Things

考虑以下几个方面：

- 操作系统本身所提供的开发工具
- 操作系统向硬件接口移植的难度
- 操作系统的内存要求
- 可裁剪性
- 是否提供硬件的驱动程序
- 操作系统的实时功能

# 编程语言的选择

**C/C++**

**Java**

**汇编语言**

选哪个好  
呢？



考虑因素：

- 通用性
- 可移植性程度
- 执行效率
- 可维护性

# 集成开发环境考虑的因素

- 系统调试器的功能
- 支持库函数
- 编译器开发商是否持续升级编译器
- 连接程序是否支持所有的文件格式和符号格式

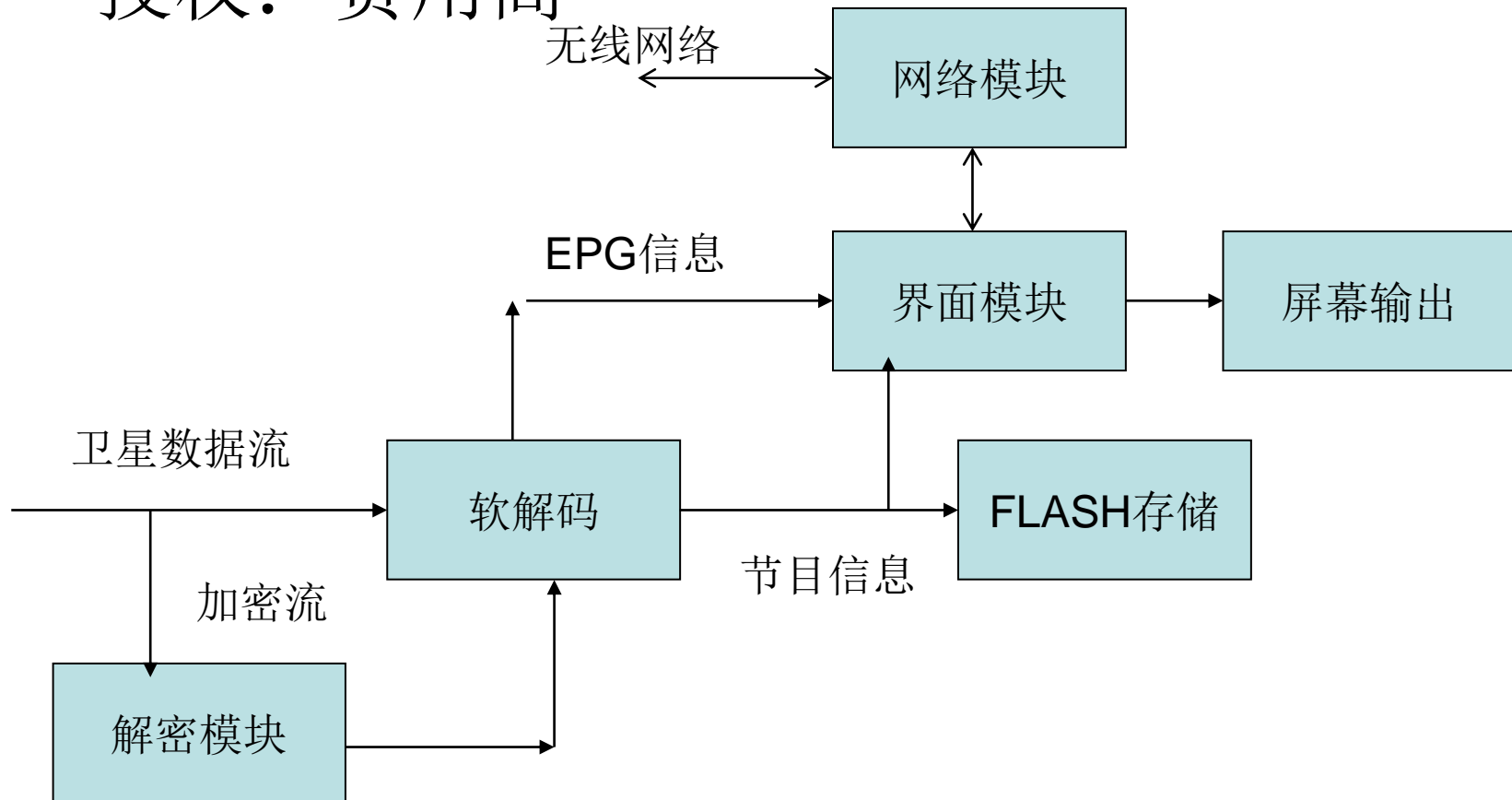
# 硬件调试工具的选择

## 常用的硬件调试工具

- 实时在线仿真器(ICE, In-Circuit Emulator)
- 驻留监控软件
- ROM仿真器
- JTAG仿真器

# 软件组件的选择

- 免费： 往往开源,但可靠性需要考虑。
- 授权： 费用高



# 硬件/软件设计阶段

- 硬件设计

- 设计硬件子系统（**top-down**方法）
- 分成模块；设计框图。例如：
- **CPU**子系统、存储器子系统、硬件接口系统等。
- 元器件选择
- 线路板设计与制做
- 线路板测试



# 硬件/软件设计阶段

- 软件设计
  - 设计软件子系统
  - 定义软件接口：模块接口、函数接口。
  - 引导与操作系统移植
  - 驱动程序设计
  - 应用程序设计与调试

嵌入式软件的开发主要采用的是“宿主机-目标机”的交叉开发模式。常见的软件开发步骤如下所示：

（1）配置开发环境及**BSP**开发。选择合适的开发工具，针对嵌入式的硬件环境对操作系统进行设置剪裁，另外增加**BSP**支持。

（2）编写用户程序和简单仿真调试。建立交叉编译开发环境，开发用户程序，将其下载到目标板上调试，应用程序开发完毕后，和文件系统一起次年改成文件系统的镜像文件，然后通过仿真工具对系统进行仿真和调试。

（3）系统的下载和脱机运行。当仿真完成后，评价系统功能，如果达到开发目标，则可把最终形成的文件下载并运行。

# 系统集成与测试

- 系统集成

- 把系统的软件、硬件和执行装置集成在一起，进行调试，发现并改进设计过程中的错误。

- 系统集成测试

- 系统的集成测试是将开发的硬件系统、软件系统和其他相关因素综合起来，对整个产品进行的全面测试。
- 常见的测试方法有离线单板硬件测试和综合测试两种方法。

## 传统软硬件设计过程的基本特征：

- 系统在设计一开始就被划分为软件和硬件两大部分
- 软件和硬件独立进行开发设计
- “Hardware first” approach often adopted

## 隐含的一些问题：

- 1、硬件设计时缺乏对软件任务实现的清晰了解，硬件设计时就具有一定的盲目性。
- 2、硬件实现时，通常选用专用功能的芯片或模块来构建硬件平台，很难发挥软硬件的综合优势。
- 3、系统的设计错误只有到系统具体实现后，进行集成、调试时才能发现，影响开发效率。
- 4、系统整体优化时，通常只能改善软件的性能，而无法充分地改善硬件的资源。
- 5、系统集成相对滞后，NRE（Non-Recurring Engineering cost）大

# 传统设计过程中的尖锐矛盾

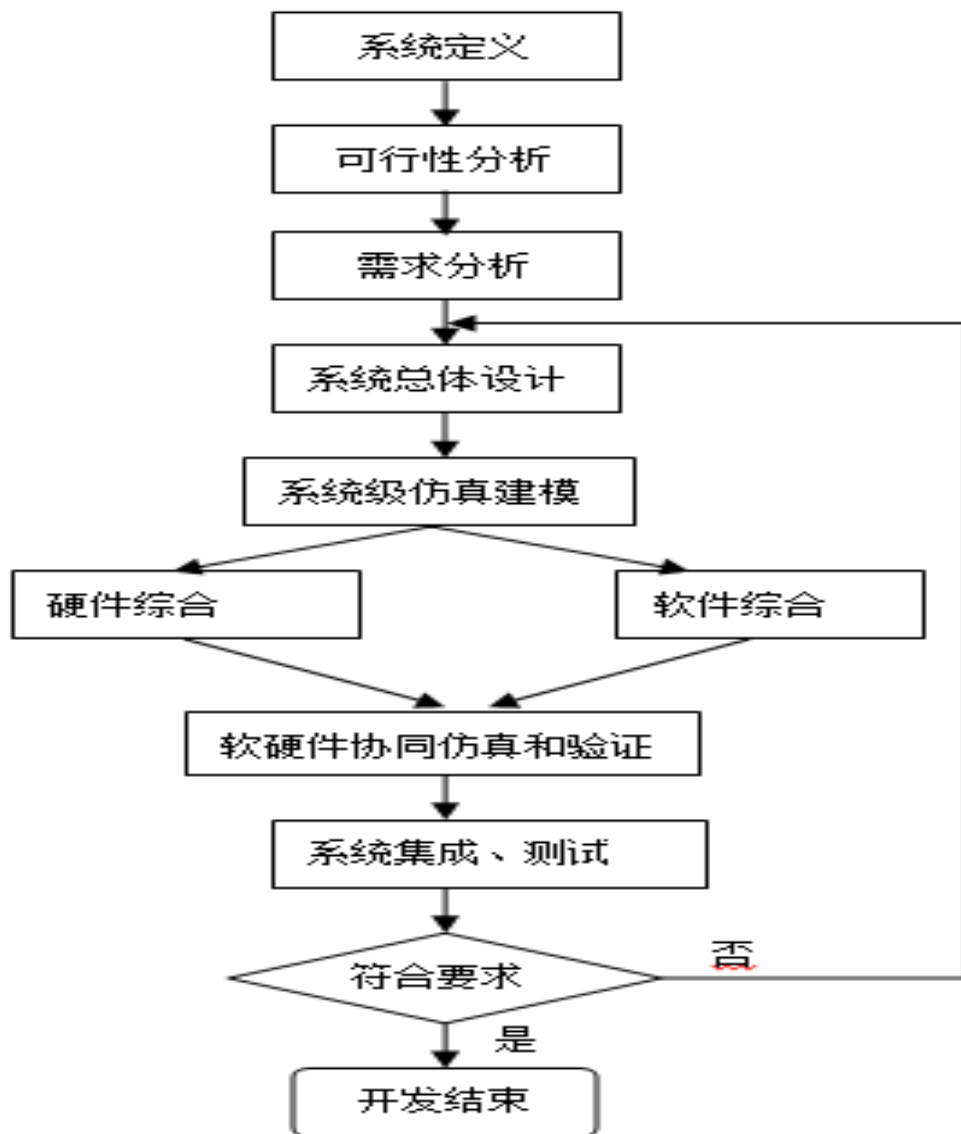
- 随着设计复杂程度的提高，软硬件设计中的一些错误将使开发过程付出昂贵的代价
- “**Hardware first**” approach often compounds (混合) software cost because software must compensate for (补偿) hardware inadequacies (不充分)

# 软硬件技术发展对嵌入式系统设计的影响

- 硬软件设计的趋势——融合、渗透
  - 硬件设计的软件化
    - VHDL, Verilog
    - HANDL-C
  - 软件实现的硬件化
    - 各种算法的ASIC
- 对系统设计的影响——协同设计
  - 增加灵活性

## 11.1.2 协同设计概念的嵌入式系统设计方法

是在系统目标要求的指导下，通过综合分析系统软硬件功能及现有资源，协同设计软硬件体系结构，找到一个软硬件功能划分的最佳点，从而使系统性能最优。



# 软件硬件协同设计的设计流程

- ◆采用方法诸如有限状态机、统一化的规格语言（CSP、HDL等）对软硬件统一描述；
- ◆对软硬件实现进行功能划分，分别用语言进行设计并将其综合起来进行功能验证和性能预测等仿真确认（协调模拟仿真）；
- ◆如果结果 达不到要求，说明划分不合理。如无问题则进行软件和硬件详细设计；
- ◆最后进行系统测试。



软硬件协同设计有如下一些基本要求：

- **统一的软硬件描述方式。**这要求软硬件支持统一的设计和分析工具及技术，并允许在一个集成环境中仿真和评估系统软硬件设计，并且支持系统任务在软硬件之间相互移植。
- **交互式软硬件划分技术。**这要求允许不同的软硬件划分设计进行仿真和比较，并需要辅助最优化决策及应用实施。
- **完整的软硬件模型基础。**这要求设计过程的每个阶段都必须支持评价，并支持阶梯式的开发方法与软硬件整合。
- **正确的验证方法。**

目前一些厂商已提供了协同设计的集成化平台或者模型：

- **ARM ESL平台：**

**ARM ESL**（**electronic system-level**）虚拟平台是采用了嵌入式系统协同设计方法的典型平台。该平台利用**SystemC**模型构建整个**SoC**系统。

- **RTSM**，实时系统模型，能提供快速、准确的指令仿真，用户可以**ARM**提供的**RTSM**上进行快速软件仿真。

这些平台让软件开发在更早的阶段开展，而不必等到在硬件平台上进行这些工作，这样一来软硬件开发工作可以并行提高，缩短产品上市时间，软硬件协同开发还可以尽早发现系统**bug**，降低开发风险和成本。

# 11.2

Part Two

## 基于ARM的嵌入式WEB服务器设计

---

**Web 服务器本质是一个软件，通常在 PC 机或者工作站上运行。**

**嵌入式 Web 服务器是指将 Web 服务器引入到现场测试和控制设备中，在相应的硬件平台和软件系统的支持下，使传统的测试和控制设备转变为以底层通信协议，Web 技术为核心的基于互联网的网络测试和控制设备。嵌入式 Web 服务器采用的是 B/S （Browser/Server）结构。**

## 11.2.1 系统环境搭建

系统平台的搭建主要进行了两方面的工作：

一是基于 **ARM** 的嵌入式硬件平台的构建。

二是嵌入式软件平台的构建。这部分工作主要分为三个部分：

①移植开发**bootloader** 作为系统引导程序，这里使用的是 **superboot** 作为本系统的 **bootloader**；

②移植 **Linux** 内核到硬件平台，采用 **Linux** 内核版本为 **Linux-3.0.8**；

③开发移植嵌入式平台上各外设驱动。

## 11.2.2 Web服务器原理

从功能上来讲，Web服务器监听用户端的服务请求，根据用户请求的类型提供相应的服务。用户端使用Web浏览器和Web服务器通信，Web服务器在接收到用户端的请求后，处理用户请求并返回需要的数据，这些数据通常以格式固定、含有文本和图片的页面出现在用户端浏览器中，浏览器处理这些数据并提供给用户。

# 1. HTTP协议

HTTP（超文本传输协议）协议是Web服务器与浏览器通信的协议，HTTP协议规定了发送和处理请求的标准方式，规定了浏览器和服务端之间传输的消息格式及各种控制信息，从而定义了所有Web通信的基本框架。

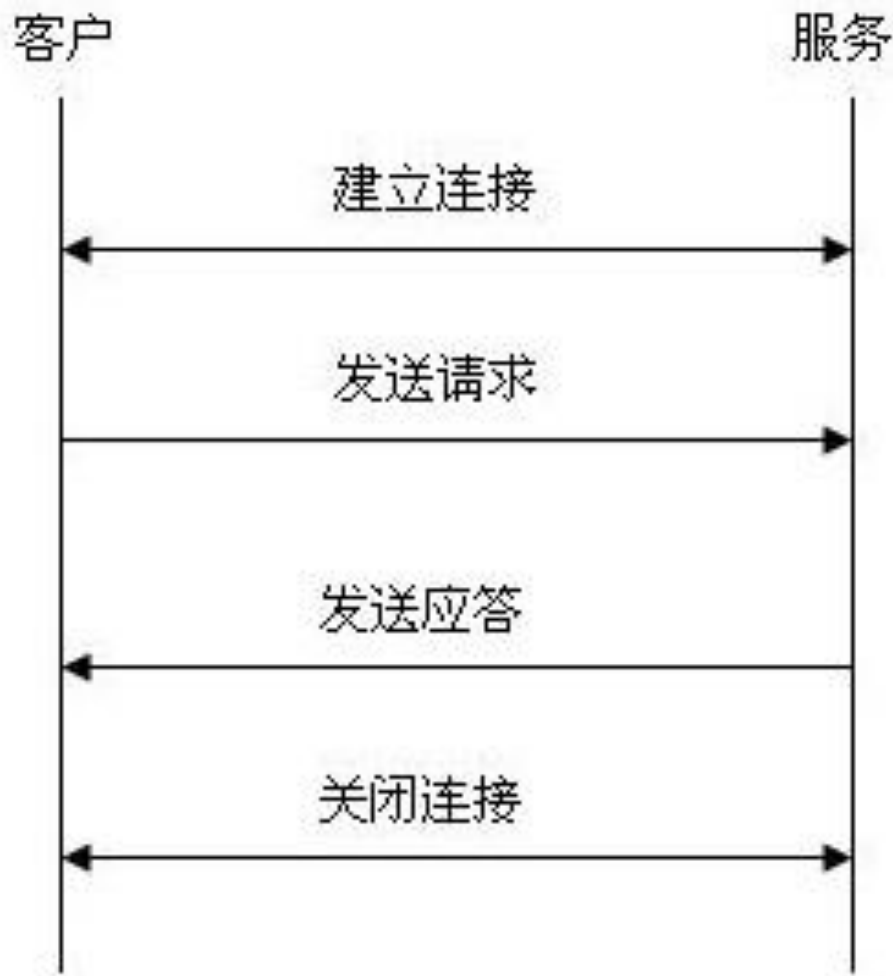
一个完整的HTTP事务由以下4个阶段组成：

（1）客户与服务器建立TCP连接；

（2）客户向服务器发送请求；

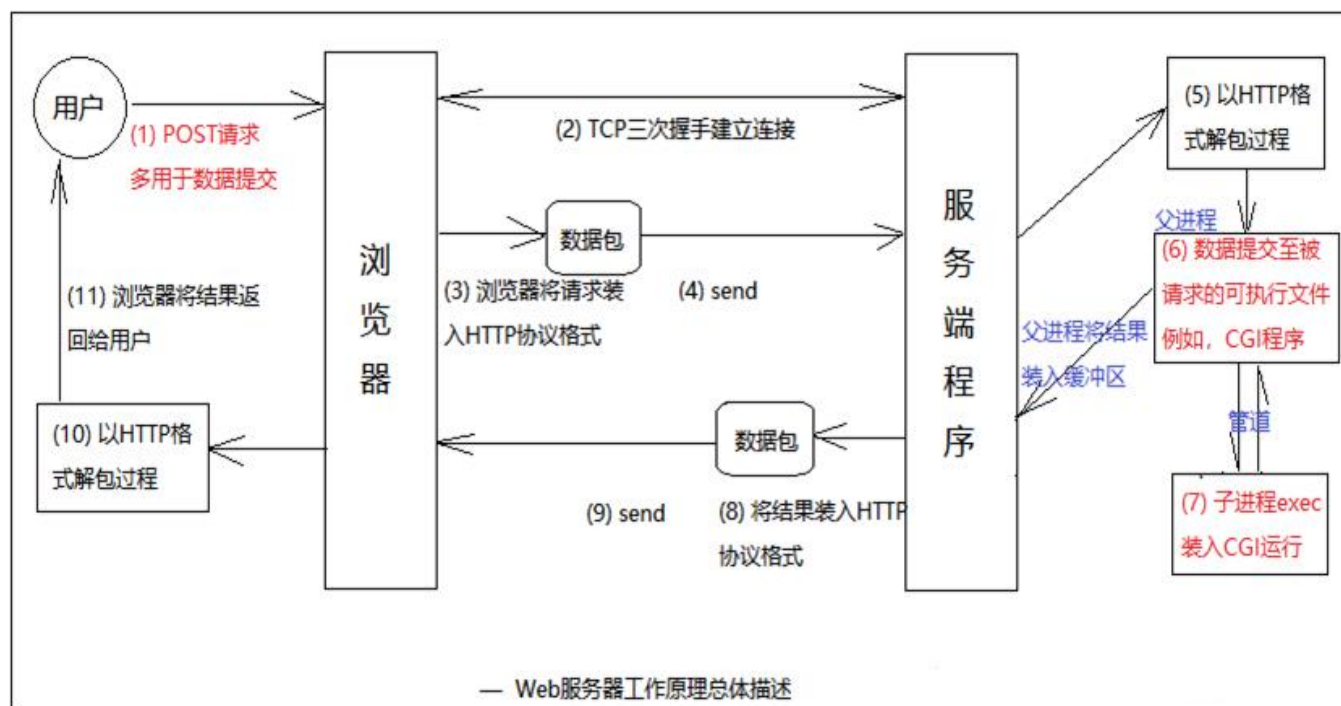
（3）如果请求被接受，则由服务器发送应答，在应答中包括状态码和所要的文件（一般是HTML文档）；

（4）客户与服务器关闭连接。



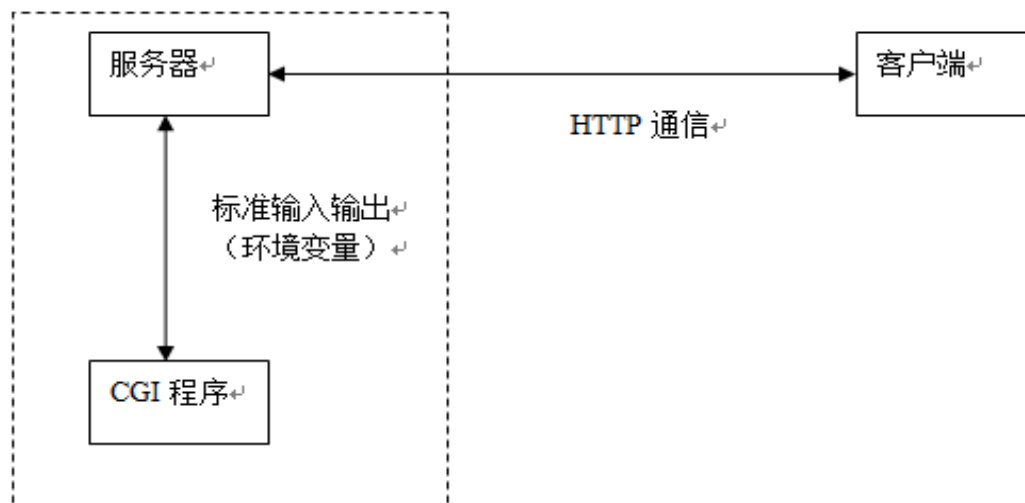


## Web服务器之数据提交工作原理图



# 通用网关接口(CGI)

- **CGI**: 通用网关接口 (Common Gateway Interface) 是一个Web服务器主机提供信息服务的标准接口。通过**CGI接口**，**Web服务器**就能够获取客户端提交的信息，转交给服务器端的**CGI程序**进行处理，最后返回结果给客户端。
- 组成**CGI**通信系统的是两部分：一部分是html页面，就是在用户端浏览器上显示的页面。另一部分则是运行在服务器上的**CGI程序**。



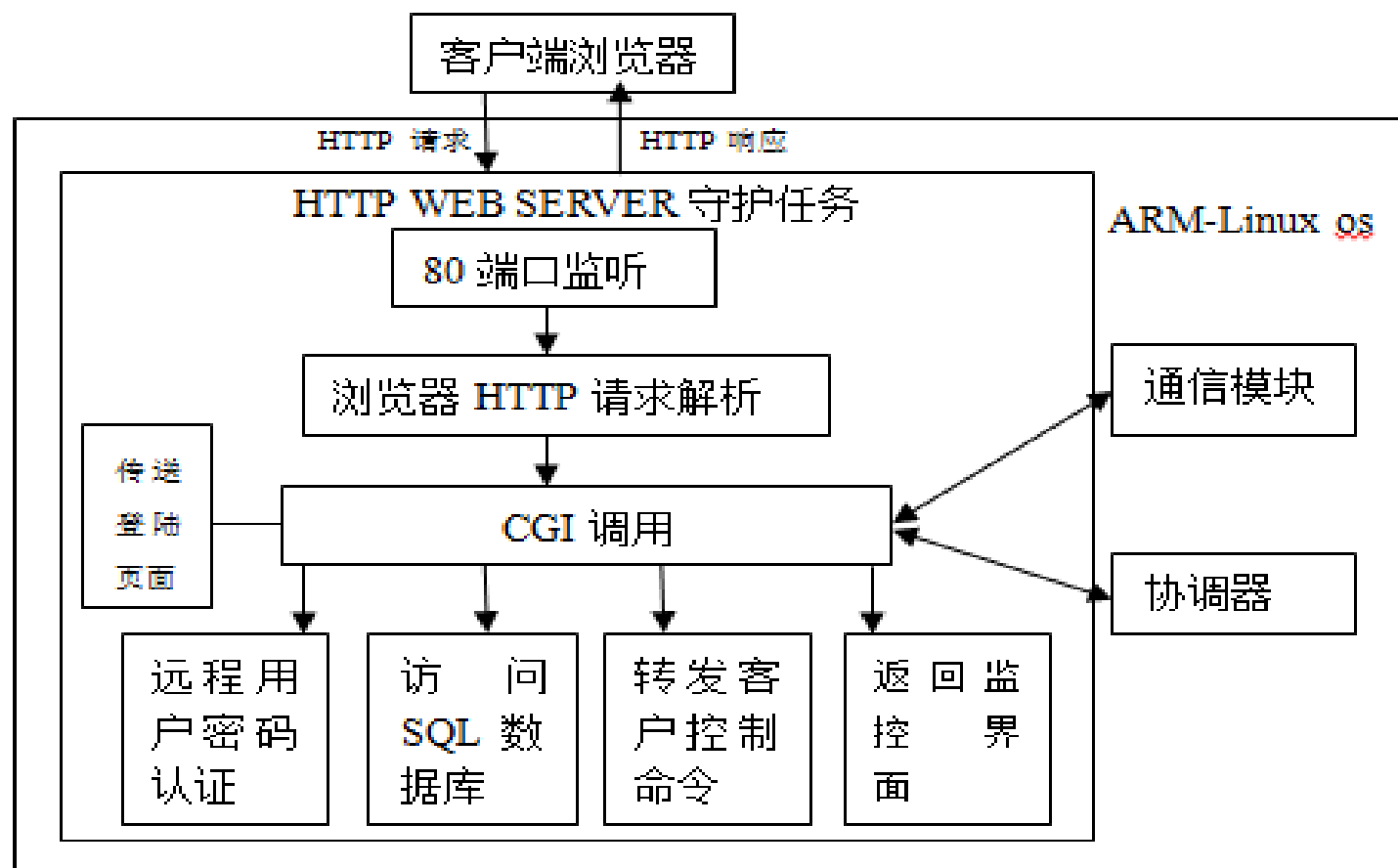
## Servlet与CGI的比较

- 1.当用户浏览器发出一个Http/CGI的请求，或者说调用一个CGI程序的时候，服务器端就要新启用一个进程(而且是每次都要调用)，调用CGI程序越多，就要消耗系统越多的处理时间。而Servlet充分发挥了服务器端的资源并高效的利用。每次调用Servlet时并不是新启用一个进程，而是在一个Web服务器的进程中共享和分离线程，而线程最大的好处在于可以共享一个数据源，使系统资源被有效利用。
- 2.传统的CGI程序，不具备平台无关性特征，系统环境发生变化，CGI程序就要瘫痪，而Servlet具备Java的平台无关性，在系统开发过程中保持了系统的可扩展性、高效性。
- 3.传统技术中，一般大都为二层的系统架构，即Web服务器+数据库服务器，导致网站访问量大的时候，无法克服CGI程序与数据库建立连接时速度慢的瓶颈，从而死机、数据库死锁现象频繁发生。而我们的Servlet有连接池的概念，它可以利用多线程的优点，在系统缓存中事先建立好若干与数据库的连接，到时候若想和数据库打交道可以随时跟系统”要”一个连接即可。

## 11.2.3 嵌入式WEB服务器设计

### 1. 嵌入式WEB服务器的工作流程

一个经典的嵌入式WEB服务器系统软件主要由HTTP WEB Server守护任务模块、CGI程序和外部通信模块3部分组成。



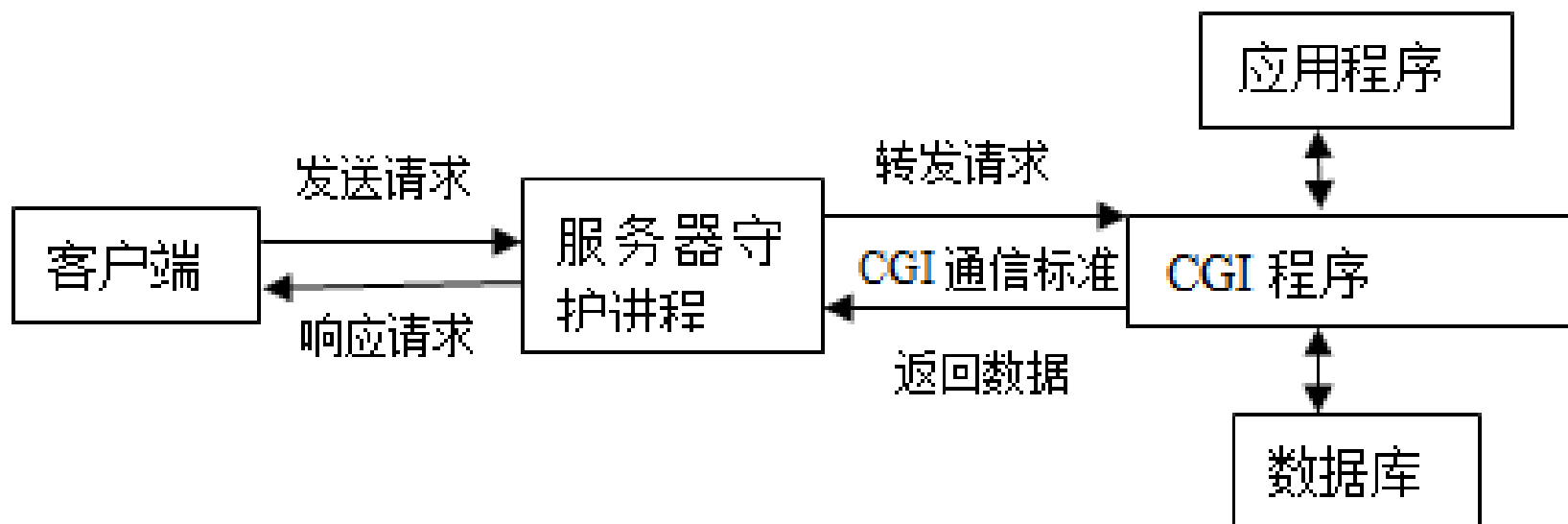
## 2.嵌入式WEB服务器选择

ARM+Linux下主要有三个Web 服务器：httpd、thttpd 和 boa。Httpd 是最简单的一个Web服务器，它的功能最弱，不支持认证，不支持CGI。Thttpd 和boa都支持认证，都支持CGI等，但是boa的功能更全，应用范围更广。因此这里通过移植boa Web服务器来实现嵌入式Web服务器功能。

CGI程序通常分为以下两部分：

- 根据POST方法或GET方法从提交的表单中接收数据。
- 用printf()函数来产生HTML源代码， 并将经过解码后的数据正确地返回给浏览器。

### 3. CGI 程序设计



客户端与服务器通过CGI标准接口通信示意图

CGI程序主要分为以下几部分:

(1) 接收客户端提交的数据。

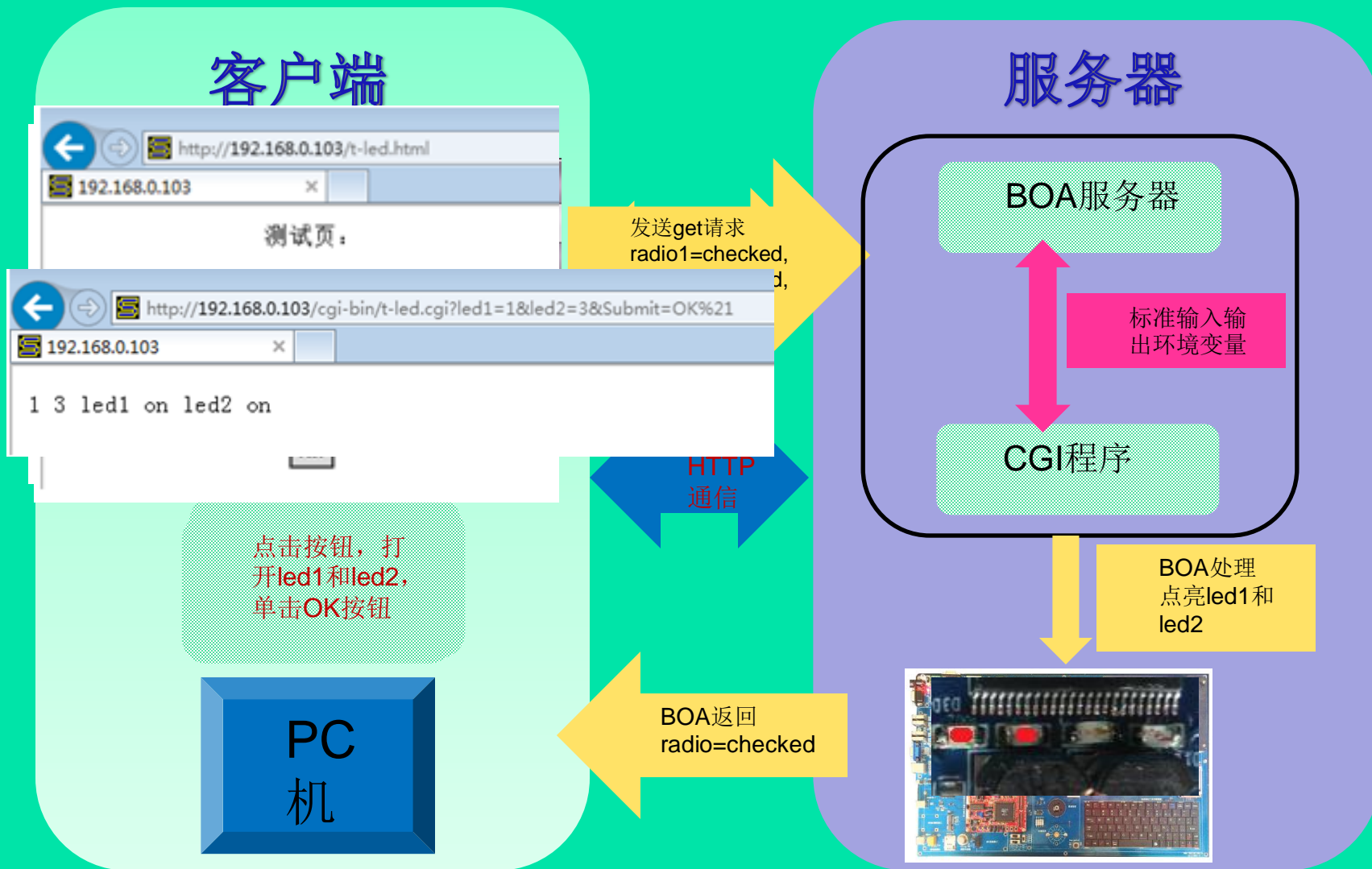
以GET方法提交数据，则客户端提交的数据被保存在QUERY\_STRING 环境变量中，通过调用函数getenv("QUERY\_STRING")来读取数据。

(2) URL编码的解码。

解码即编码的逆过程。在程序中，只要对于由①所述方法提取的数据进行URL 编码逆操作，就可以得到客户端传过来的数据。最后将解析出来的name/value保存在一个自定义的结构体中。

(3) 根据上一部分解析出来的变量/ 值对，判断客户端请求的含义，利用Linux下进程间通信机制传送消息给相应的应用程序主进程，

# BOA的工作原理





# Boa WebServer的下载

- <http://www.boa.org> 下载 `boa-0.94.13.tar.gz`

Boa Webserver



Larry Doolittle and Jon Nelson

- [News!](#) (last updated 23 February 2005)
  - Latest Released Version (0.94.13) [here](#) (signature [here](#))
  - Latest Development Version (0.94.14rc21) [here](#) (signature [here](#))
  - Read the CHANGES file [here](#).
  - [Documentation](#)
  - [Screenshot](#)
  - [Some Recent Benchmarks](#)
- More recent versions of Boa have been benchmarked using zb (ZeusBen)
- [Public Key for jnelson@boa.org \[Key ID 78E2F518\]](#) Also available via
- The key fingerprint is:

# Boa编译

1. 把boa-0.94.13.tar.gz放到虚拟机/opt
2. `#tar xvzf boa-0.94.13.tar.gz`
3. `#cd boa-0.94.13`
4. `#cd src`
5. 利用configure 工具配置生成Makefile 文件  
`#. /configure`
6. 修改生成的Makefile 文件的第31行、第32行（设置交叉编译器），将

```
CC = gcc  
CPP = gcc -E
```

修改为:

```
CC = arm-linux-gcc  
CPP = arm-linux-gcc -E
```

# Boa编译

7. 可修改defines.h 文件的第30行

将**#define SERVER\_ROOT “/etc/boa”**

可修改为:**#define SERVER\_ROOT “/gec/web”**

该处定义的是WEB 服务器的文件根目录，跟boa.conf 文件中的DocumentRoot 一致即可。

8. 修改compat.h的第120行

```
120 #define TIMEZONE_OFFSET(foo) foo##->tm_gmtoff
121 #else
122 #define TIMEZONE_OFFSET(foo) timezone
123 #endif
124
125 #ifdef HAVE_TM_ZONE
126 #define TIMEZONE(foo) foo##->tm_zone
```

修改成**#define TIMEZONE\_OFFSET(foo) (foo)->tm\_gmtoff**

防止在make 时出现如下错误提示：

util.c:100:1: pasting “t” and “->” does not give a valid preprocessing token。

# Boa编译

## 9. 修改boa.c文件

注释掉下面句话（第**210**行至第**215**行）：

```
#if 0
if (passwdbuf == NULL) {
DIE("getpwuid");
}
if (initgroups(passwdbuf->pw_name, passwdbuf->pw_gid) == -1) {
DIE("initgroups");
}
#endif
```

否则会出现错误： **getpwuid: No such file or directory**

注释掉下面语句：

```
#if 0
if (setuid(0) != -1) {
DIE("icky Linux kernel bug!");
}
#endif
```

否则会出现错误： **- icky Linux kernel bug!: No such file or directory**

# Boa编译

## 10. 编译boa

**#make**

至此，在**src** 目录中将得到交叉编译后的**boa** 程序。

# 修改BOA配置信息

BOA的配置信息都保存在文件boa.conf中，故该文件是BOA的配置文件，该文件是最终要放在实验箱的/gec/web目录下，BOA默认在该路径下读取相关的所有配置信息。

该文件目前在虚拟机的/opt/boa-0.94.13。

下面修改boa.conf文件：

## 1、修改用户与用户组信息

第48行：User nobody

第49行：Group nogroup

改为：**第48行： User 0**

**第49行： Group 0**

在根文件系统中的/etc/passwd 文件中没有nobody 用户，所以设成0。

在根文件系统中的/etc/group 文件中没有nogroup 组，所以设成0。

# 修改BOA配置信息

2、相关日志文件存放位置项，保留将保存日志文件，根据需要可以选择是否注释掉

第62行: **Errorlog /var/log/boa/error\_log**

改为: **#Errorlog /var/log/boa/error\_log**

第74行: **Accesslog /var/log/boa/access\_log**

改为: **#Accesslog /var/log/boa/access\_log**

3、第94行，打开ServerName 的设置：

**将#ServerName www.your.org.here 前面的#号去掉**，该项默认为未打开，执行Boa 会异常退出，提示“gethostbyname::No such file or directory”，所以必须打开。

# 修改BOA配置信息

4、第111行: `DocumentRoot /var/www`

改为: `DocumentRoot /www`

注意: 实验箱的目录/`www`存放网页文件

5、把“第130行: `DirectoryMaker /use/lib/boa/boa_indexer`”注释掉

6、第155行, 将`MimeTypes /etc/mime.types`

修改为: `MimeTypes /www/mime.types`

7、第160行, 将`DefaultType text/plain`

修改为: `DefaultType text/html`

8、第188行, 将`Alias /doc /usr/doc` 注释掉。



# 修改BOA配置信息

9、ScriptAlias 的设置：

第193行，将ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/

修改为：

ScriptAlias /cgi-bin/ /www/cgi-bin/

至此，BOA 服务器配置已经完成。

- 测试BOA主要分为3步：编写测试页面，启动web服务器，执行测试。
- 根据配置文件中的信息可知，测试页面放在开发板根目录/www下。

### html部分内容说明，具体查看html教程

- <html> 与 </html> 之间的文本描述网页
- <body> 与 </body> 之间的文本是可见的页面内容
- <h1> 与 </h1> 之间的文本被显示为标题
- <p> 与 </p> 之间的文本被显示为段落
- <a> 与 </a> 是通过 <a> 标签进行定义HTML 链接
- <body> 元素定义了 HTML 文档的主体。

## ■ 1、编写测试页面index.html如下：

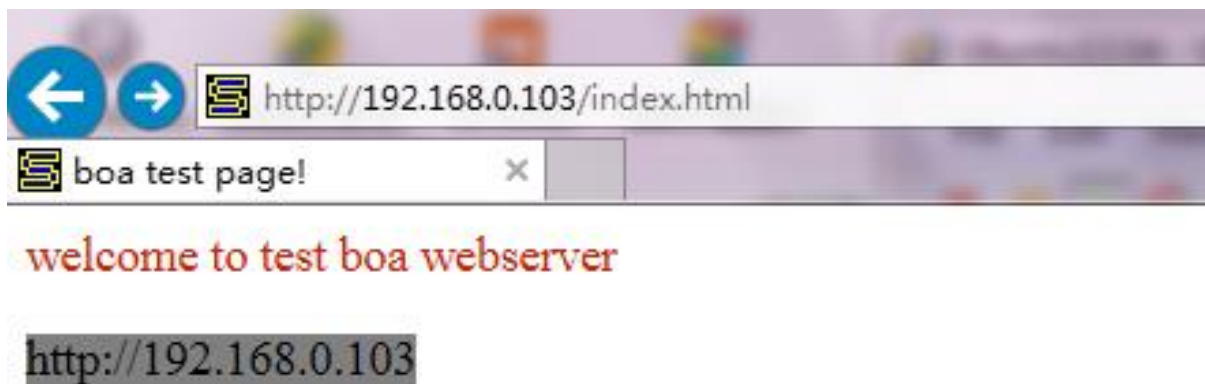
```
1 <html>
2   <title>
3     boa test page!
4   </title>
5   <head>
6     <font color="#cc2200"><b></b>welcome to test boa webserver</font><p>
7   </head>
8   <body>
9     It is the homepage of testing boa webserver <p>
10    <font style="background-color:#808080">http://192.168.0.103</font><p>
11  </body>
12 </html>
13 |
```

## ■ 2、启动web服务器

- 1) 把生成的boa文件、配置文件boa.conf、index.html放到目标机的/www目录下。
- 2) 在目标机上
  - #ps |grep boa (寻找包含boa字符的进程，此时会寻找到目标机中已经有运行的boa，是/www/boa)
  - #kill 125(boa process)
  - #cd /www
  - #./boa

### ■ 3、执行测试

- 若实验箱的ip为192.168.0.103，故windows系统中的ip改为与实验箱同一网段，若为192.168.0.100。
- 开发板与主机连好网线
- Windows系统中打开“运行” → “cmd” → “ping 192.168.0.103”
- 若网络ping通，打开IE，输入：  
“http://192.168.0.103/index.html”
- 结果如下：



1、虚拟机上编写测试代码hello.c，该测试文件内容为打印“Hello World.”，代码如下：

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    printf("Content-type:text/html\n\n");
    printf("<html>\n");
    printf("<head><title>CGI Output</title></head>\n");
    printf("<body>\n");
    printf("<h1>Hello World.</h1>\n");
    printf("</body>\n");
    printf("</html>\n");
    exit(0);
}
```

2、编译测试程序

```
arm-none-linux-gnueabi-gcc -o hello.cgi hello.c
```

3、把编译生成的hello.cgi文件放到实验箱/[www/cgi-bin](#)目录下。

4、打开Windows系统的IE浏览器，输入：“http://192.168.0.103/cgi-bin/hello.cgi”

- 5、运行结果如下：

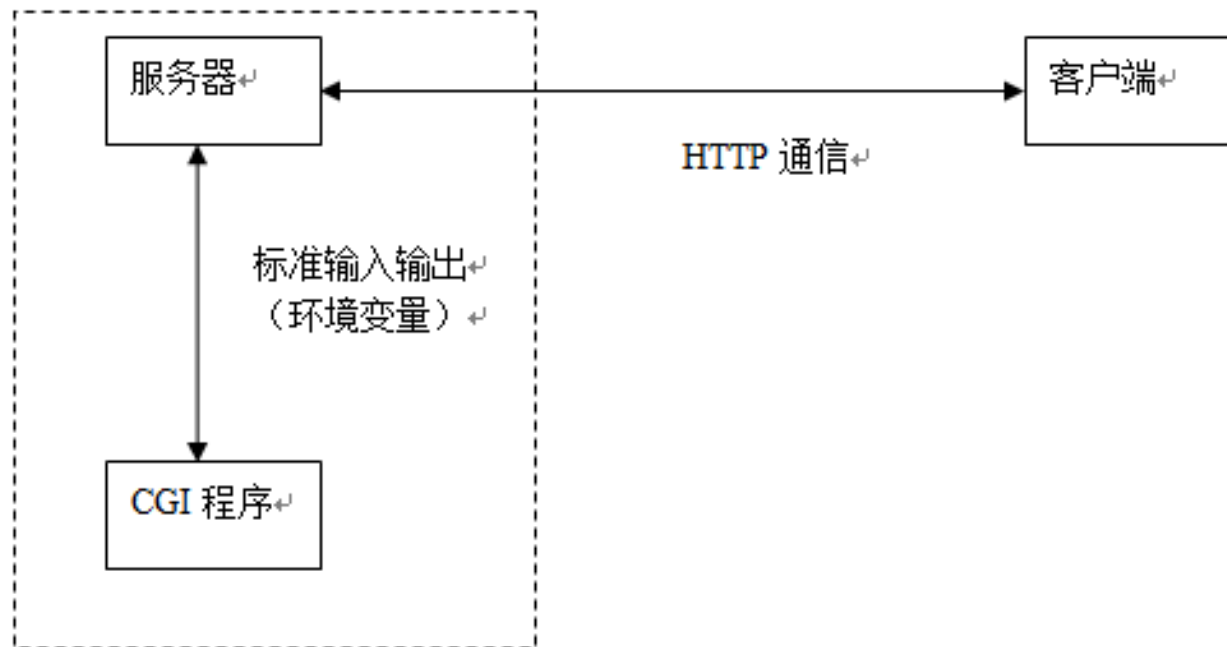


Hello World.

# BOA的运用



# 通用网关接口(CGI)



- 服务器和客户端之间的通信，是客户端的浏览器和服务器端的 **http** 服务器之间的 **HTTP** 通信，我们只需要知道浏览器请求执行服务器上哪个 **CGI** 程序就可以了，其他不必深究细节，因为这些过程不需要程序员去操作。
- 服务器和 **CGI** 程序之间的通讯才是我们关注的。一般情况下，服务器和 **CGI** 程序之间是通过标准输入输出来进行数据传递的，而这个过程需要环境变量的协作方可实现。

# 通用网关接口(CGI)

- CGI 程序通过两种方式来获取浏览器发送过来的表单数据：  
**QUERY\_STRING** 环境变量和**标准输入**。
- 两种方式分别对应于浏览器的“GET”和“POST”两种提交表单的方式。
- 当浏览器使用 **GET** 方式来提交表单时，**CGI** 程序必须通过 **QUERY\_STRING** 环境变量来获取表单内容。
- 当浏览器使用 **POST** 方式提交表单时，**CGI** 程序必须通过标准输入获取表单内容，同时，可以通过**CONTENT\_LENGTH** 环境变量来获取到表单内容的大小。

# 实例1、test.html和test.c

## ■ test.html代码如下：

```
1 <p align="center" class="STYLE4"> 测试页： </p>
2 <div align="center">
3   <form id="form1" name="form1" method="get" action="/cgi/bin/test.cgi">
4     <p>
5       <label>
6         input your first name:
7         <input type="text" name="name" value="text" />
8       </label>
9     </p>
10    <p>
11      <label>
12        <input type="submit" name="Submit" value="OK !" />
13      </label>
14    </p>
15  </form>
16 </div>
17
```

# 实例1、test.html和test.c

## ■ test.c代码如下：

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 int main(void)
5 {
6     char *get1;
7     char *get2;
8     char *get3;
9
10    printf("Content-type:text/html\n\n");
11    /*
12     getenv()用来取得参数envvar环境变量的内容。QUERY_STRING 环境变量中则包含了所有表单的内容。
13     用法:char *getenv(char *envvar);
14     函数说明:getenv()用来取得参数envvar环境变量的内容。参数envvar为环境变量的名称,
15     如果该变量存在则会返回指向该内容的指针。环境变量的格式为envvar=value。
16     getenv函数的返回值存储在一个全局二维数组里,当你再次使用getenv函数时不用担心会覆盖上次的调用结果。
17     返回值:执行成功则返回指向该内容的指针,找不到符合的环境变量名称则返回NULL。
18     */
19    get1 =getenv("QUERY_STRING");    //当输入"abc"时, get1指向字符串:"name=abc&submit=OK+%21"
20    if(get1 == NULL)
21        printf("failed get data!\n");
22    /*
23     strtok()用来将字符串分割成一个个片段。原型:char *strtok(char *s, char *delim); 参数s指向欲分割的字符串,
24     参数delim则为分割字符串中包含的所有字符。当strtok()在参数s的字符串中发现参数delim中包含的分割字符时,
25     则会将该字符改为\0 字符。在第一次调用时, strtok()必需给予参数s字符串,往后的调用则将参数s设置成NULL。
26     每次调用成功则返回指向被分割出片段的指针。
27     */
28    get2=strtok(get1,"=&");    //get2指向字符串:"name"
29    get3=strtok(NULL,"=&");    //get3指向字符串:"abc"
30    printf("<br/>");
31    printf("*****");
32    printf("<br/>");
33    printf("*                hello %s!                *",get2);
34    printf("<br/>");
35    printf("*****");
36    printf("<br/>");
37 }
38
```

# 代码解析

- **getenv()函数:**
- **getenv()**用来取得参数`envvar`环境变量的内容。`QUERY_STRING`环境变量中则包含了所有表单的内容。
- **用法:**`char *getenv(char *envvar);`
- **函数说明:**`getenv()`用来取得参数`envvar`环境变量的内容。参数`envvar`为环境变量的名称，如果该变量存在则会返回指向该内容的指针。环境变量的格式为`envvar=value`。`getenv`函数的返回值存储在一个全局二维数组里，当你再次使用`getenv`函数时不用担心会覆盖上次的调用结果。
- **返回值:**执行成功则返回指向该内容的指针，找不到符合的环境变量名称则返回`NULL`。

# 代码解析

- **strtok()**函数:
- **原型**: `char *strtok(char *s, char *delim);`
- **功能**: 分解字符串为一组标记串。**s**为要分解的字符串，**delim**为分隔符字符串。
- **说明**: 首次调用时，**s**必须指向要分解的字符串，随后调用要把**s**设成**NULL**。**strtok**在**s**中查找包含在**delim**中的字符并用**NULL('\0')**来替换，直到找遍整个字符串。返回指向下一个标记串。当没有标记串时则返回空字符**NULL**。

# 使用 POST 方式

- 当浏览器使用 POST 方式提交表单时，CGI 程序必须通过标准输入获取表单内容，同时，可以通过CONTENT\_LENGTH 环境变量来获取到表单内容的大小。例如，将上面的网页中的 method="get"修改为 method="post"，那么 cgi 的程序需要变为：
- `const char *slen = getenv("CONTENT_LENGTH");` // 获取 POST 方式提交的表单内容大小
- `int len = atoi(slen);` // 将字符串形式的长度转换为整数
- `char *query = (char *)malloc(len + 1);` // 分配空间，用来保存表单内容
- `memset(query, 0, len + 1);` // 将分配到的空间的内容清空
- `read(0, query, len);` // 读取表单内容
- // 这里，query 字符串应该等于  
"username=gec&password=123456&Submit=提交"其中，read()函数用来读取文件中的数据，它包含三个参数：第一个参数表示文件序号，0 即为标准输入文件；第二个参数为保存读取到的数据的缓冲区；第三个参数为读取的长度。

- 1) 把生成的boa文件、配置文件boa.conf放到目标机的/ged/web目录下。
- 2) 把test.c放到ubuntu中进行交叉编译，生成test.cgi二进制文件；把test.html放到目标机的/www目录下，把test.cgi放到目标机的/www/cgi-bin目录下
- 3) 在目标机上

**#ps |grep boa** (寻找包含boa字符的进程，此时会寻找到目标机中已经有运行的boa，是/ged/web/boa)

**#kill 125**

**#cd /www**

**#./boa**



- 4) 若实验箱的ip为192.168.0.103, 故windows系统中的ip改为与实验箱同一网段, 若为192.168.0.100。
- 5) 开发板与主机连好网线
- 6) Windows系统中打开“运行” → “cmd” → “ping 192.168.0.103”
- 7) 若网络ping通, 打开IE, 输入:  
“http://192.168.0.103/test.html”
- 结果如下:

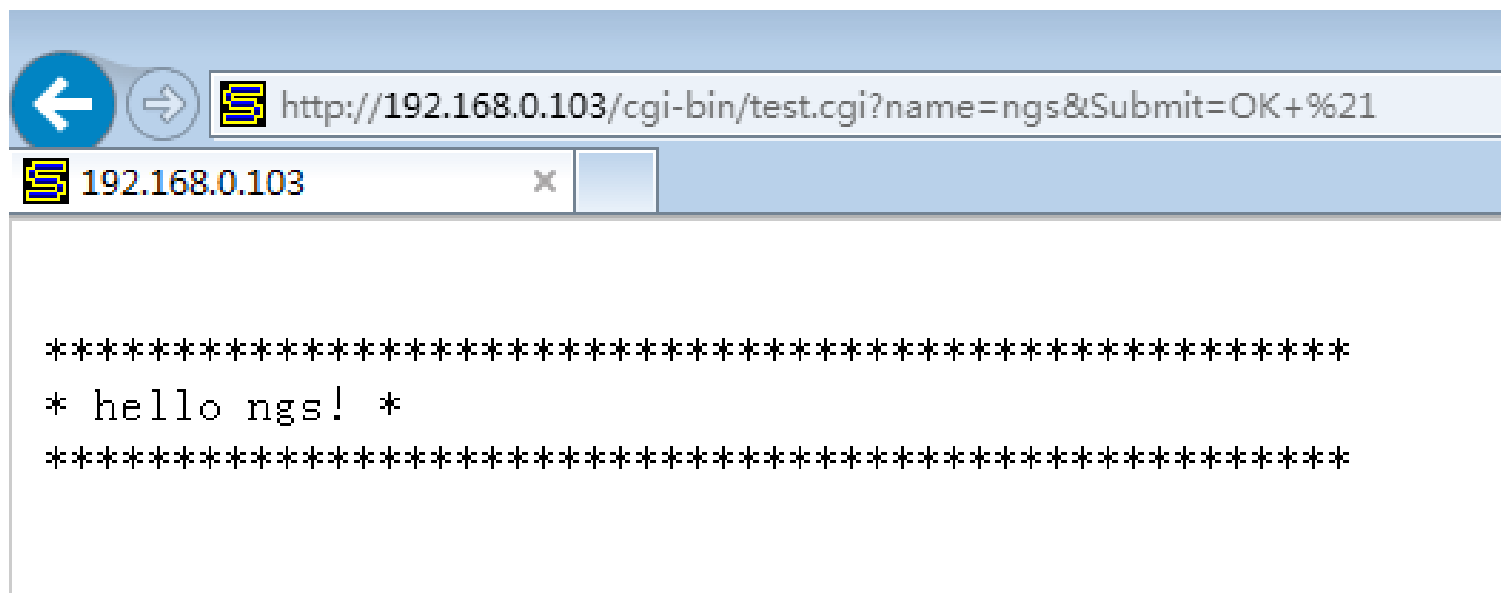


测试页:

input your first name:

OK !

- 在文本框输入字符串，如：“ngs”，单击OK按钮，结果如下：



# 实例2、config.html和config.c

## ■ config.html代码如下：

```
1 <HTML>
2 <HEAD>
3 <META http-equiv="content-type" content="text/html; charset=utf-8" />
4 <TITLE> Gec BOA Test Page </TITLE>
5 </HEAD>
6 <BODY>
7 <FORM action="/cgi-bin/config.cgi" method="get">
8 Username:<input type="text" name="username"><br />
9 Password:<input type="password" name="password"><br />
10 <input type="submit" name="Submit" value="OK">
11 </FORM>
12 </BODY>
13 </HTML>
```

# 实例2、config.html和config.c

## ■ config.c代码如下：

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 int main(void)
6 {
7     char *env_data = NULL;
8     const char *result;           // 定义一个字符串指针，用来保存结果
9     char username[100], password[100];
10    printf("Content-type:text/html\n\n");
11    env_data = getenv("QUERY_STRING");
12    printf("Content-type:text/html\n\n");
13    if (env_data == NULL)
14    {
15        printf("failed get data!");
16    }
17    else
18    {
19        sscanf(env_data, "username=%[^&]&password=%[^&]", username, password);
20        // 从 env_data 表单内容中分离 username 和 password
21        // if((strcmp(username, "gec") == 0)&& (strcmp(password, "123456") == 0))
22        if((strcmp(password, "123456") == 0))
23            result = "Login OK";           // 登录成功
24        else
25            result = "Login Failed";       // 登录失败
26        printf("<p>%s</p>\n", result);      // 用 H1 标题样式输出登录结果
27    }
28    return 0;
29 }
30
31
```

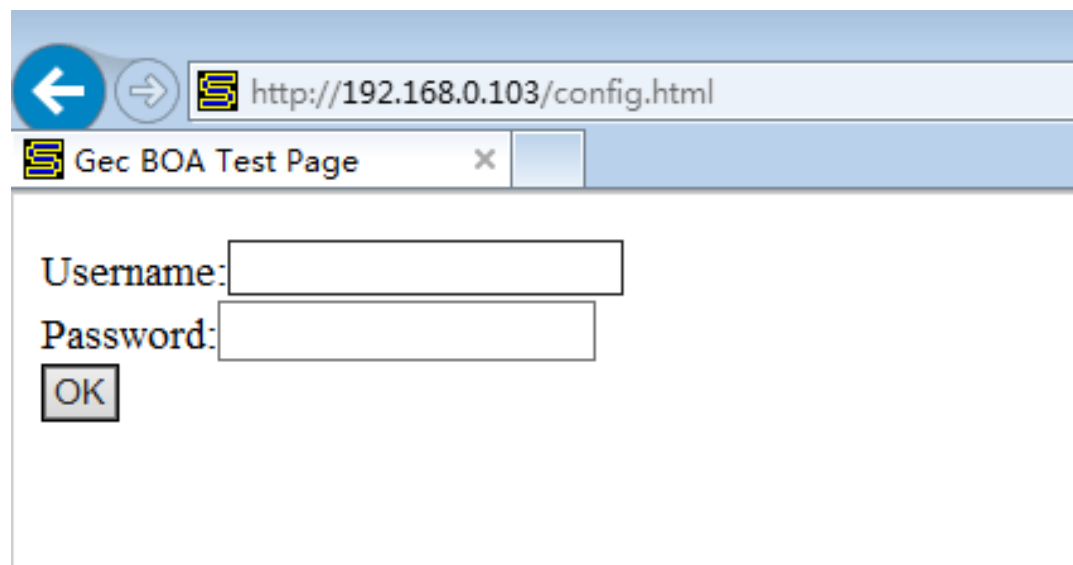
# 代码解析

- **sscanf()**函数：
- 功能：从一个字符串中读进与指定格式相符的数据。
- 函数原型：
  - `int sscanf( const char *, const char *, ...);`
  - `int sscanf(const char *buffer,const char *format,[argument ]...);`
  - `buffer`存储的数据
  - `format`格式控制字符串
  - `argument` 选择性设定字符串
  - `sscanf`会从`buffer`里读进数据，依照`format`的格式将数据写入到`argument`里。
  -

- 操作步骤如同测试实例1一样;
- 输入:

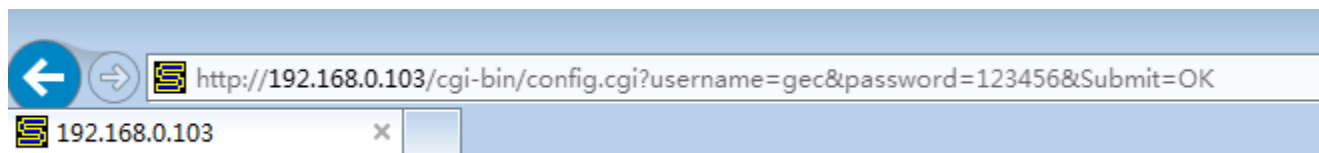
<http://192.168.0.103/config.html>

运行结果如下:

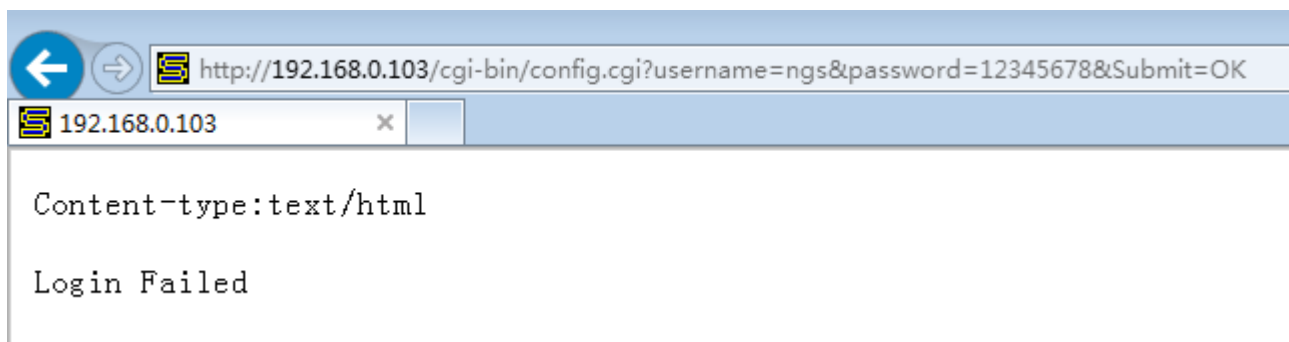


The screenshot shows a web browser window with a blue header bar. The address bar contains the URL `http://192.168.0.103/config.html`. Below the address bar, there is a tab titled "Gec BOA Test Page". The main content area displays a login form with two input fields: "Username:" and "Password:". Below these fields is an "OK" button.

- 当输入账号： **gec**， 密码： **123456**， 结果如下：



- 当输入账号： **ngs**， 密码： **12345678**， 结果如下：



实验：下面代码中是通过网页实现对试验箱中**led1**和**led2**的亮灭控制。



# t-led.html和t-led.c

## ■ t-led.html代码如下：

```
1 <p align="center" class="STYLE4"> 测试页: </p>
2 <div align="center">
3   <form id="form1" name="form1" method="get" action="/cgi-bin/t-led.cgi">
4     <p>
5       打开led1
6         <input type="radio" name="led1" value="1" checked />
7         <br />
8       关闭led1
9         <input type="radio" name="led1" value="2" />
10      <br />
11
12     打开led2
13       <input type="radio" name="led2" value="3" checked/>
14       <br />
15     关闭led2
16       <input type="radio" name="led2" value="4" />
17     <br />
18   </p>
19   <p>
20     <input type="submit" name="Submit" value="OK!" />
21   </p>
22 </form>
23 </div>
24
25
```

# t-led.html和t-led.c

## ■ t-led.c代码如下:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 #include <unistd.h>
6 #include <sys/ioctl.h>
7 #include <sys/types.h>
8 #include <sys/stat.h>
9 #include <fcntl.h>
10 #include <sys/select.h>
11 #include <sys/time.h>
12 #include <errno.h>
13
14 #define LED1      0
15 #define LED2      1
16 #define LED3      2
17 #define LED4      3
18
19 #define LED_ON      1
20 #define LED_OFF     0
21
22 int main(void)
23 {
24     char *get1;
25     char *get2;
26     char *get3;
27     int cmd[2];
28
29     printf("Content-type:text/html\n\n");
30
31     get1 =getenv("QUERY_STRING");
32     //当选择2和4, get1指向字符串"led1=2&led2=4&Submit=OK%21"
33     if(get1 == NULL)printf("failed get data!\n");
```

```
34
35     get2=strtok(get1,"=&");
36     get3=strtok(NULL,"=&");
37     cmd[0]=atoi(get3);
38     printf("%d\n",cmd[0]);
39
40     get3=strtok(NULL,"=&");
41     get3=strtok(NULL,"=&");
42     cmd[1]=atoi(get3);
43     printf("%d\n",cmd[1]);
44
45     int ledn_fd,a;
46     ledn_fd = open("/dev/leds", O_RDWR);
47     if (ledn_fd < 0) {
48         perror("open device ledn_fd");
49         exit(1);
50     }
51
52     if(cmd[0]==1)
53     {
54         printf("led1 on\n");
55         ioctl(ledn_fd,LED_ON,LED1);
56     }
57     else if(cmd[0]==2)
58     {
59         printf("led1 down\n");
60         ioctl(ledn_fd,LED_OFF,LED1);
61     }
62
63     if(cmd[1]==3)
64     {
65         printf("led2 on\n");
66         ioctl(ledn_fd,LED_ON,LED2);
67     }
68     else if(cmd[1]==4)
69     {
70         printf("led2 down\n");
71         ioctl(ledn_fd,LED_OFF,LED2);
72     }
73 }
```

输入：<http://192.168.0.103/t-led.html>前，  
先在目标机中加载LED驱动模块，如图所示。

6) 加载 led 驱动：

```
# insmod /lib/modules/3.4.39/led.ko
```

- 输入：<http://192.168.0.103/t-led.html>  
运行结果如下，同时实验箱中led1和led2亮。



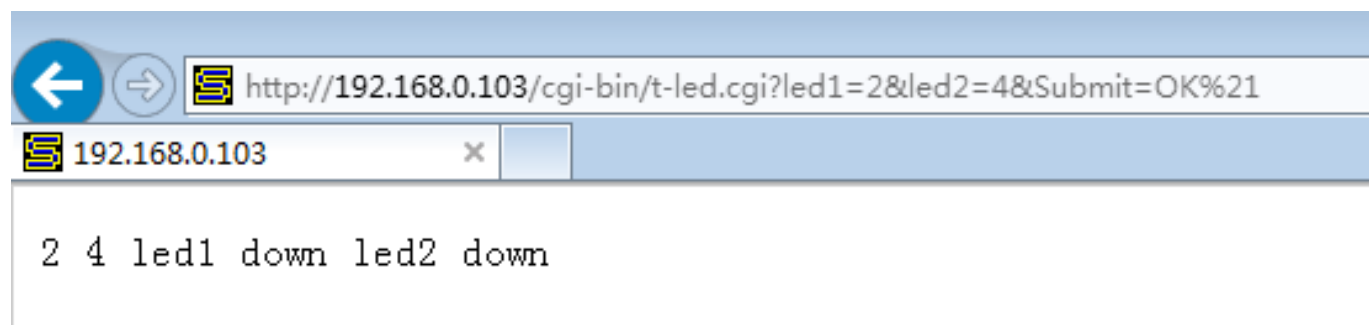
运行结果如下，同时实验箱中1ed1和1ed2灭。



测试页：

打开led1 ☐  
关闭led1 ☒  
打开led2 ☐  
关闭led2 ☒

OK!



# 11.3

Part Three

## 物联网网关设计实例

---

# 物联网

## 概念起源

麻省理工学院Ashton教授于1999年最早提出，其理念是基于射频识别技术(RFID)、电子代码(EPC)等技术，在互联网的基础上，构造一个实现**全球物品信息实时共享**的**实物互联网**，即**物联网**。

# 物联网

**互联网**是指将两台计算机或者是两台以上的计算机终端、客户端、服务端通过计算机信息技术的手段互相联系起来的結果，人们可以与远在千里之外的朋友相互发送邮件、共同完成一项工作、共同娱乐。

**移动互联网**是人依靠移动终端实现移动终端与移动终端、固定终端两两互聯，实现人与人的通信；

**物联网**是依靠固定和移动网络实现物与物之间的两两互聯，不需要人的参与就可以智能运转。

应该说，移动互联网和物联网都是互联网的延伸，只是一个向自由发展，一个向自然发展，一个是由静及动，一个是由人及物。



# 物联网的特点

## 1) 各种感知技术的广泛应用。

物联网上部署了海量的多种类型传感器，每个传感器都是一个信息源，不同类别的传感器所捕获的信息内容和信息格式不同。传感器获得的数据具有实时性，按一定的频率周期性的采集环境信息，不断更新数据。

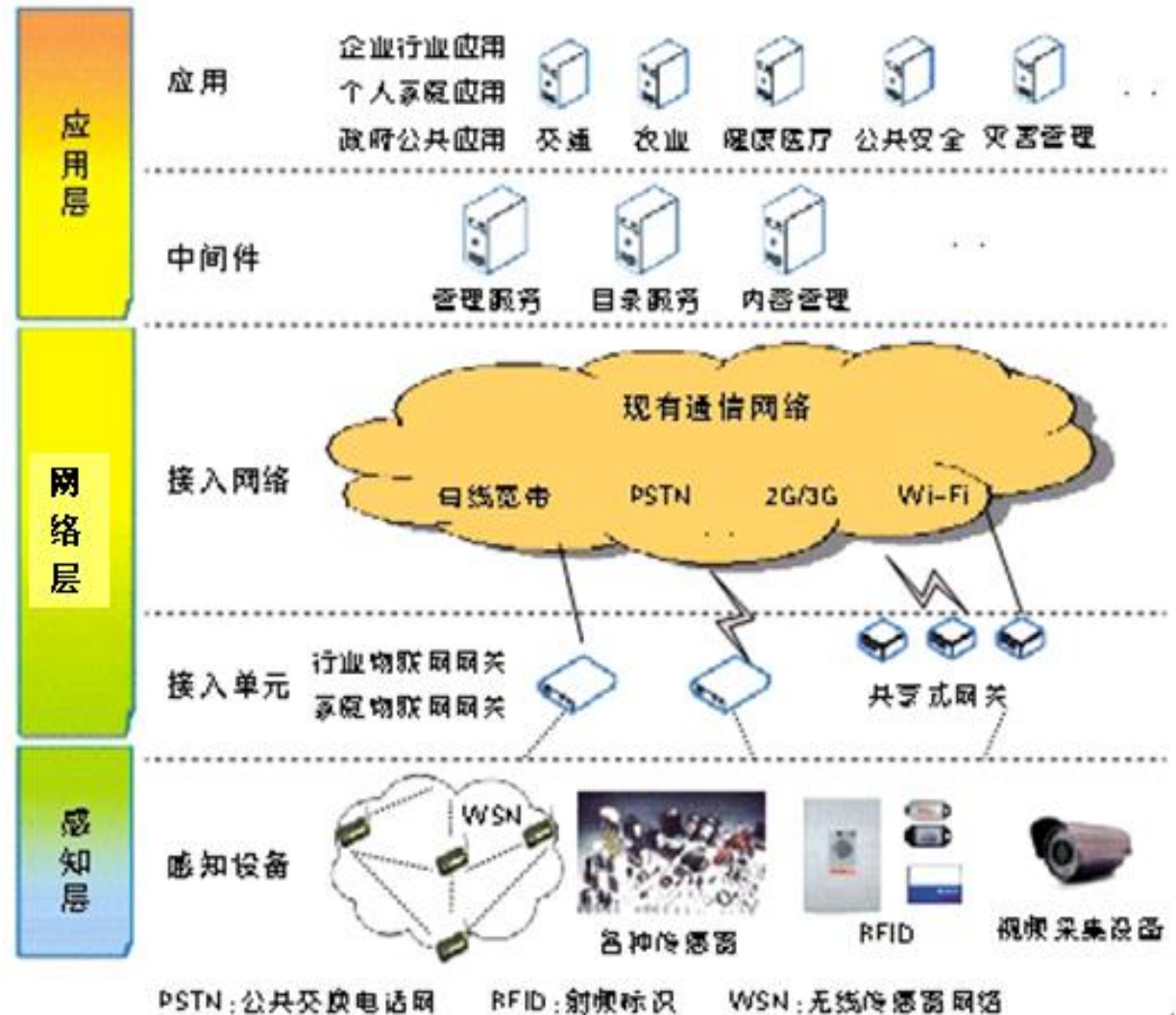
## 2) 建立在互联网上的泛在网络。

通过各种有线和无线网络与互联网融合，将物体的信息实时准确地传递出去。在传输过程中，为了保障数据的正确性和及时性，必须适应各种异构网络和协议。

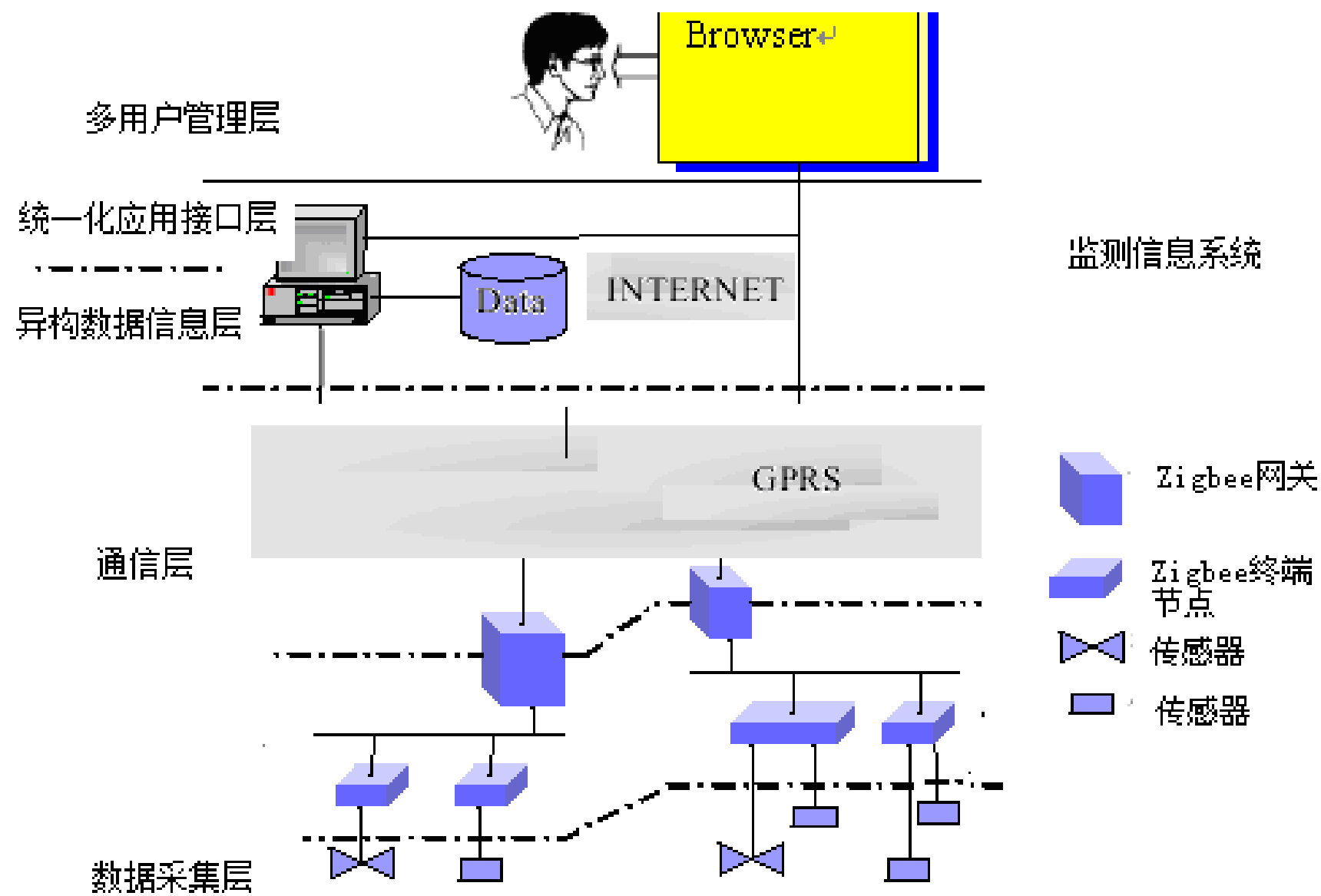
## 3) 不仅提供了传感器的连接，其本身也具有智能处理的能力，能够对物体实施智能控制。

# 物联网体系架构

物联网通常被公认为有3个层次，从下到上依次是感知层、网络层和应用层，如图所示。



11.3.1 背景介绍-环境监测系统平台整体架构



系统总体方案主要采用分层设计方法，自下而上分为数据采集层、通信层、异构数据信息层、统一化应用接口层和多用户管理层。数据采集层主要由分布在被测湿地环境中的众多**ZigBee** 终端节点组成，测量终端携带有水温、浊度、**PH**值、溶氧等多种传感器，能够实时不间断的监测湿地各种关键参数，它们和网关节点一起组成了具备高动态自组网络模式的监测体系。该检测体系具备较小的网络开销，可实现网络快速构建和数据端到端的实时传输。

整个系统采用的是查询和中断相结合的模式，大多数情况下，系统中的大部分硬件处于睡眠模式，当湿地环境发生异常时，**ZigBee** 终端节点会被唤醒，将检测环境数据和自己的节点信息如节点**ID**、电池状况等经路由计算后发送到网关节点，网关节点进行数据存储、数据预处理工作，并将数据通过**GPRS**发送到用户终端或者监控中心数据库，由监控中心数据库产生数据分析图表和报表输出。用户可通过**INTERNET**访问监控中心数据库图表系统获取实时信息。管理者也可通过**GPRS**模块和网关主动查询节点测量数据和控制节点功用。

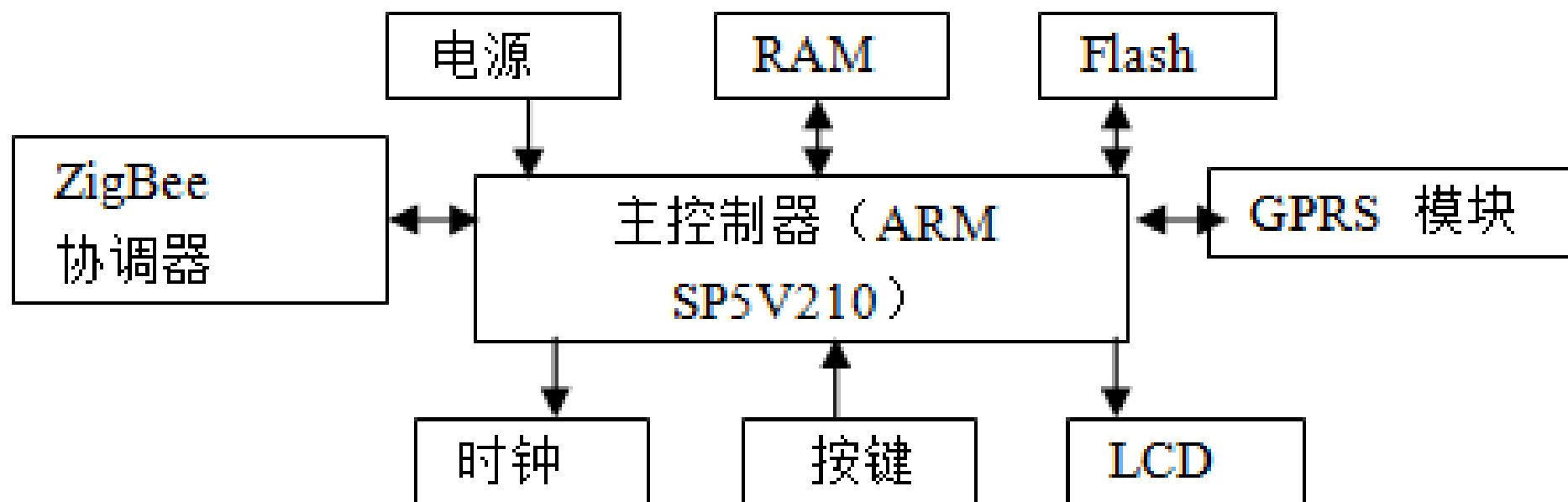
通信层完成监测系统内数据的传输，主要涵盖三个层次的传输任务：

（1）**ZigBee**协调器建立和维护**ZigBee**网络的运行，从监测终端节点接收实时监测数据，并发送到网关节点单元；

（2）工作于湿地环境中的网关节点通过**GPRS** 网络与监控中心服务器实现信息交互，能够通过**ZigBee** 协调器与众多终端节点构成网络开销小，结构动态可变化的无线自组网络。

（3）监控中心服务器对接收到的数据主要进行存储、处理和分析等工作，把接收的终端监测信息存储到数据库并可产生相应图表或者报表，另一方面亦可侦听来自因特网网络上客户端的连接，并与客户端建立套接关系；同时亦可与移动终端设备通过**WIFI**，**3G**网络完成监测数据实时通信。

### 11.3.2. 网关节点硬件设计方案



## (1) 主控制器

根据网关节点的可靠性、数据处理能力等要求，网关节点主控制器采用了**Samsung**公司基于**ARM Cortex-A8**处理器核的**S5PV210**处理器。在实际设计过程中，采用了“核心板+扩展板”的模式进行硬件平台构建。

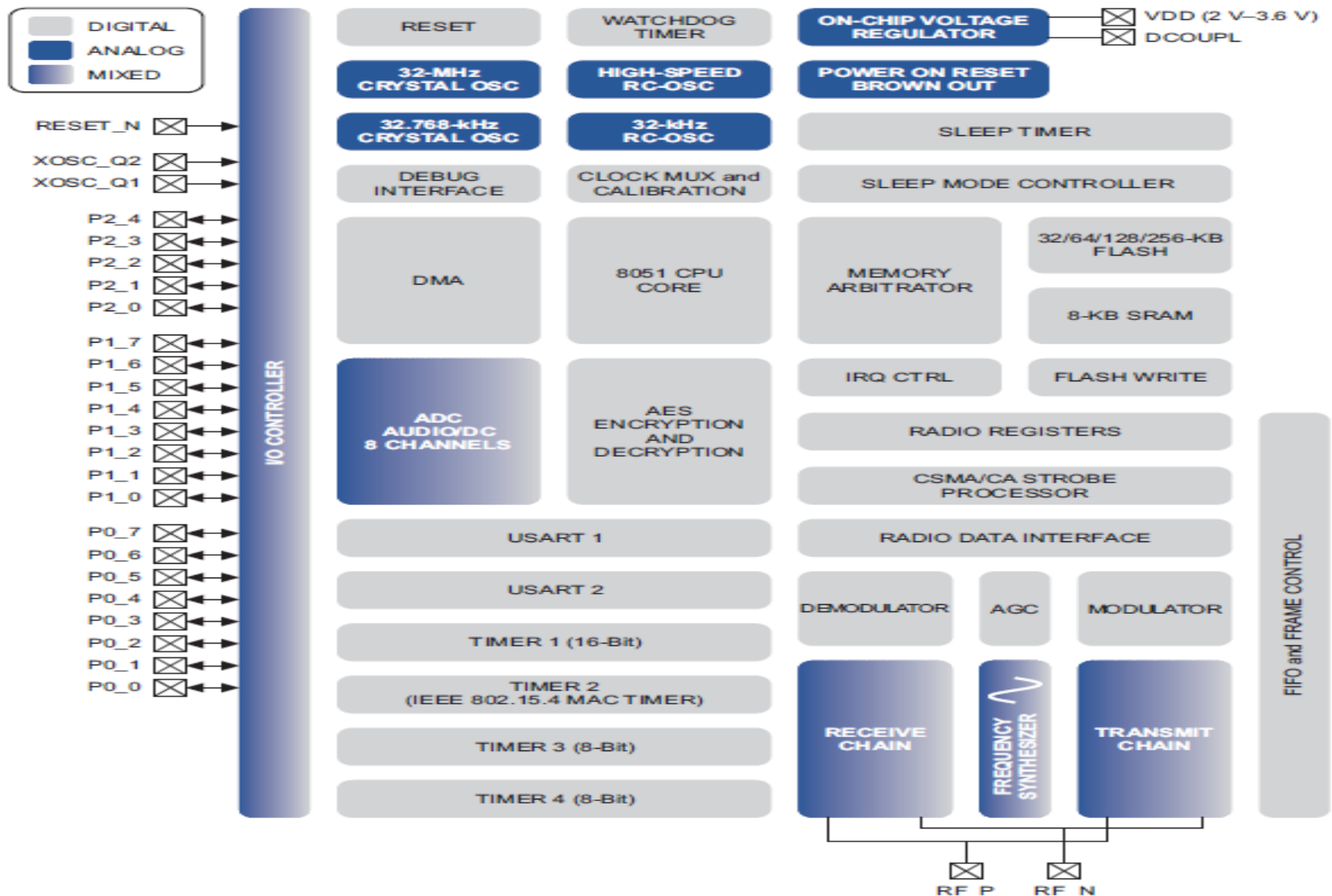


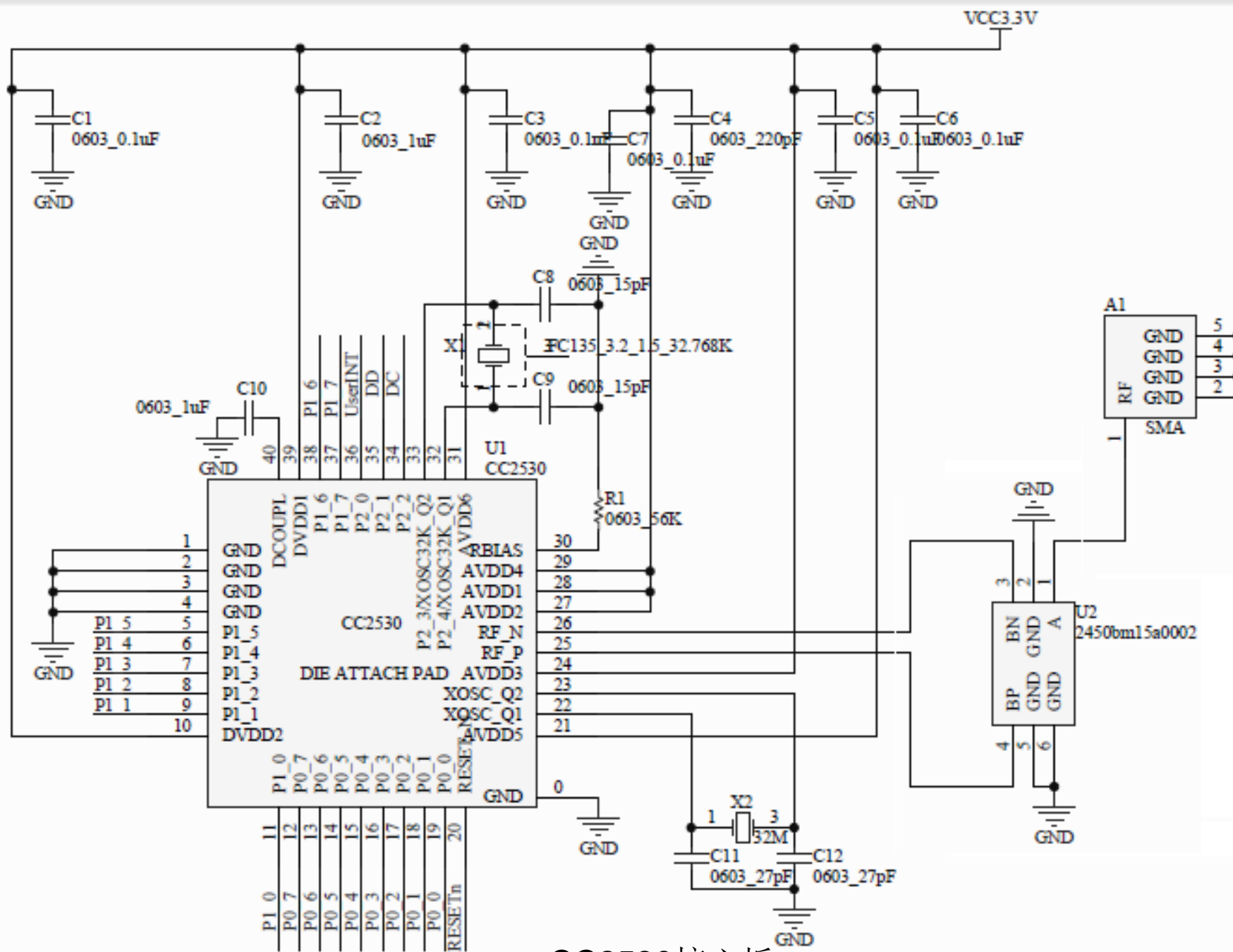
嵌入式操作系统是一种支持嵌入式系统应用的操作  
系统软件，它是嵌入式系统的极为重要的组成  
部分，通常包括与硬件相关的底层驱动软件、系  
统内核、设备驱动接口、通信协议、图形用户界  
面及标准化浏览器等。与通用操作系统相比较，  
嵌入式操作系统在系统实时高效性、硬件的相关  
依赖性、软件固化以及应用的专用性等方面有突  
出的特点。

低端应用以单片机或专用计算机为核心所构成的可编程控制器的形式存在，一般没有操作系统的支持，具有监控、伺服、设备指示等功能，带有明显的电子系统设计特点。

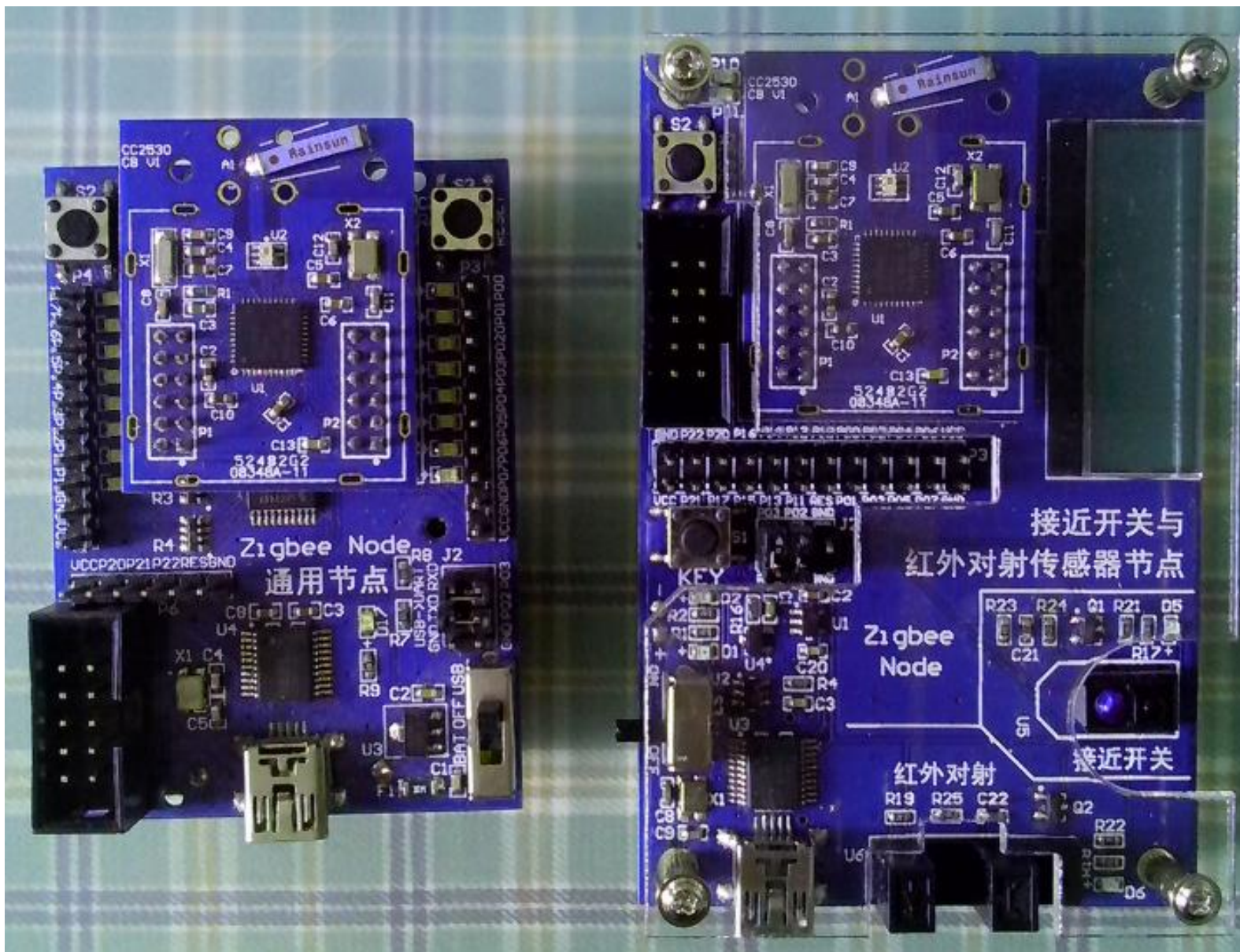
## (2) ZigBee 协调器

Zigbee节点主要分为协调器（通用节点）和传感器节点两种。基于成本和使用方便的考虑，系统采用了DRF1601作为ZigBee 协调器，它的主芯片是TI公司 CC2530F256 芯片





## CC2530核心板

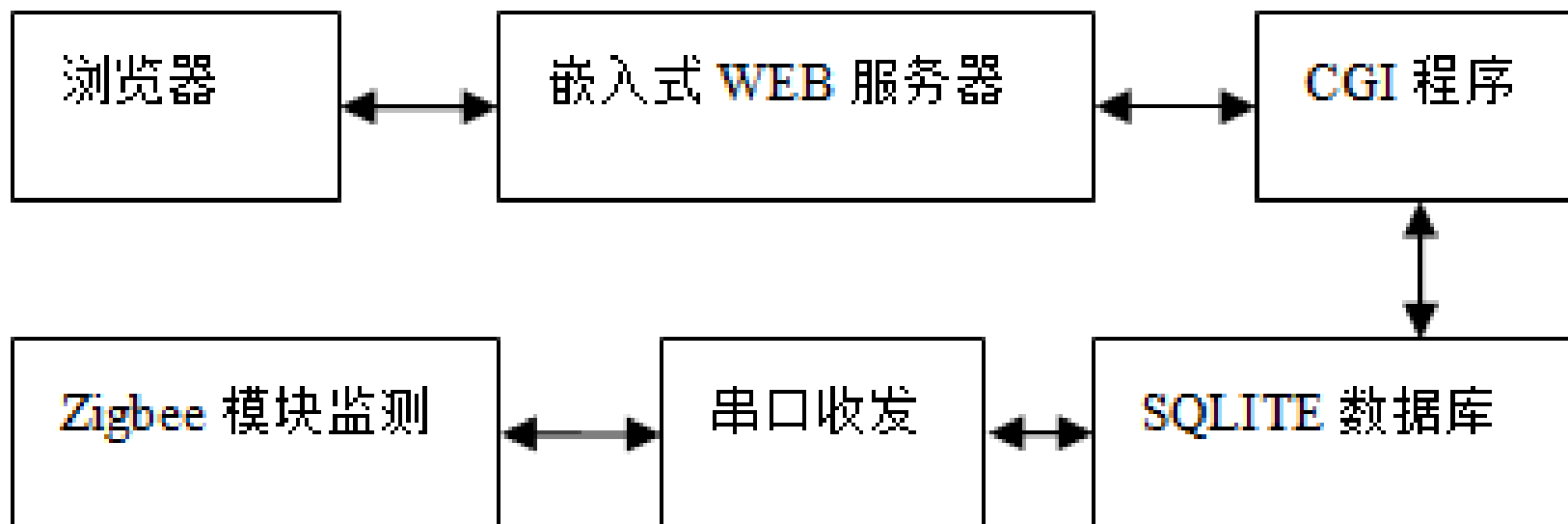


协调器（通用节点）和传感器节点实物图

## 11.3.3 系统软件设计

### 1.概述

该网关系统的应用程序分为两大块：运行在ARM-Linux平台上的嵌入式WEB服务器程序和运行在模块上的程序。

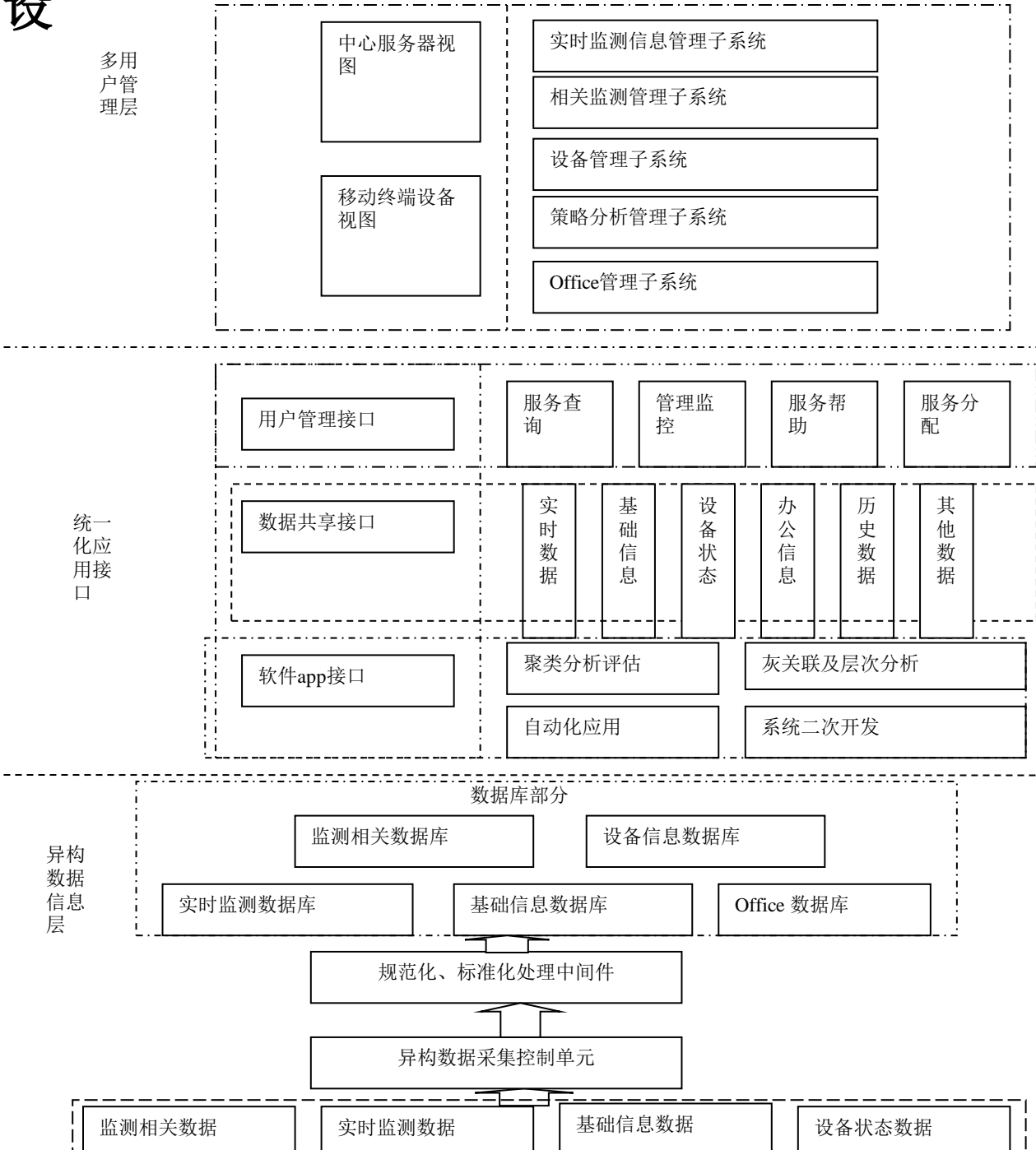


本网关使用的嵌入式 Web 服务器包括核心部分和可裁减部分，核心部分包括 HTTP 请求解析器和模块分析器。HTTP 请求解析器负责接收客户发送的 HTTP 请求报文，获得客户端信息，并把解析出来的结果保存到请求结构中；模块分析器根据配置信息调度其他模块。模块主要分为系统功能模块和用户功能模块，一旦配置了系统功能模块，该模块就对服务器收到的请求进行处理。系统功能模块主要分为3个部分：文件系统访问模块（针对静态网页）、CGI 处理模块（针对动态网页）、赋值处理（针对用户控制作用）。

**Zigbee**模块程序也分为两个部分：协调器程序部分和终端节点程序部分。这两部分也被定义为**zigbee**网络的上位机程序部分和下位机程序部分。



# 11.3.4数据库建设



根据监测数据信息量的大小，冗余性和安全性的考虑，这里将数据库建设分为两个部分：一是监控中心数据库建设，要求服务器具有较大的信息处理能力和吞吐率。这里选择**SQL SERVER**数据库作为监控中心数据开发及管理工具。二是网关节点上的**SQLite**数据库，这种数据库所对应的环境是**ARM+Linux**平台，在**B/S**结构下访问数据。网关节点上的数据库主要对信息现场及时查询负责，因而数据库规模较小，相当于监控中心服务器上数据库的简化版本。

数据库实际隶属于监测系统中的信息系统**MIS**



# 11.4

Part Four

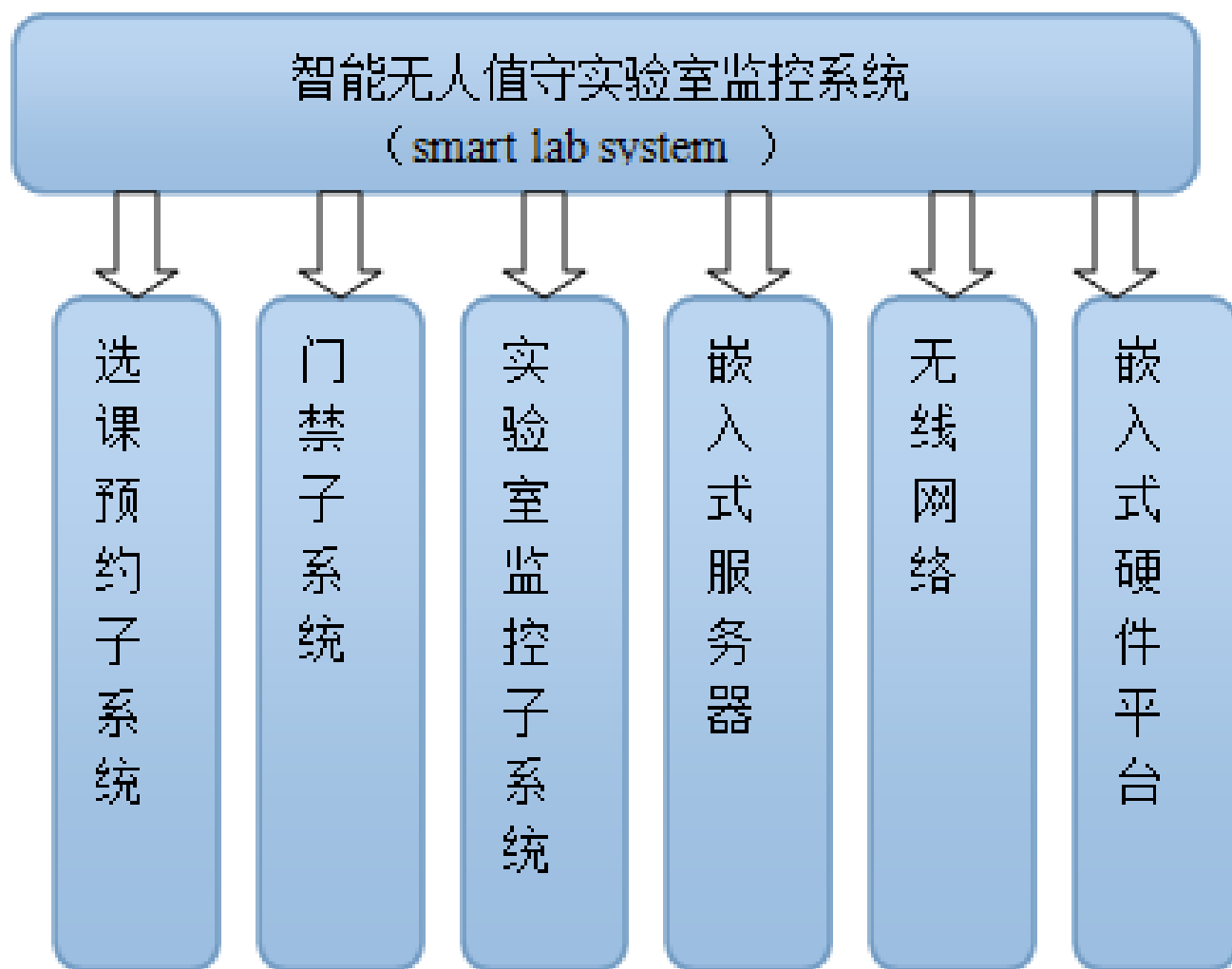
## 智能无人值守实验室监控系统设计实例

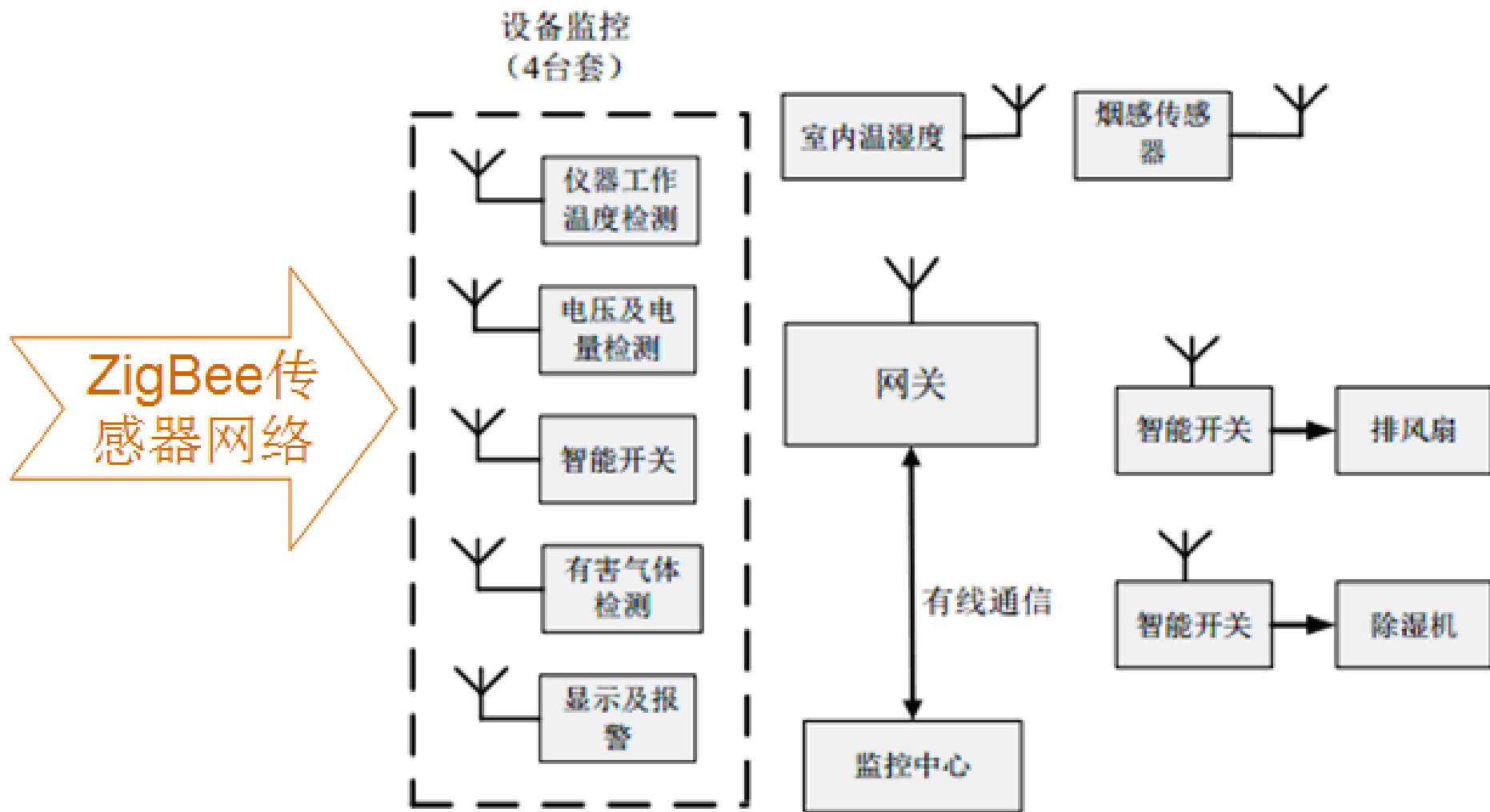
---

智能无人值守实验室监控系统具有

- (1) 可以大大降低实验室管理所需要的人力物力成本。
- (2) 可以提高实验室设备的利用率，合理规划。
- (3) 做到实验室设备的状态自动化状态监控，监管。
- (4) 无人监管实验室可以让学生自主化利用课余时间实验学习。
- (5) 成功的无人实验室系统对外推广可以产生可观的经济效益。

## 11.4.1 系统总体框架





监控子系统硬件总体原理图

实验室监控子系统是实验室智能化管理系统的一个子系统，完成实验室环境及实验设备的监控工作。包括实验室工作环境与设备运行状况等参数的检测、数据网关及异常情况的处理控制等几大功能模块。各模块采用无线通信方式，方便施工和维护。通过数据网关将采集数据上发服务器，并接收服务器下发的各种控制命令。

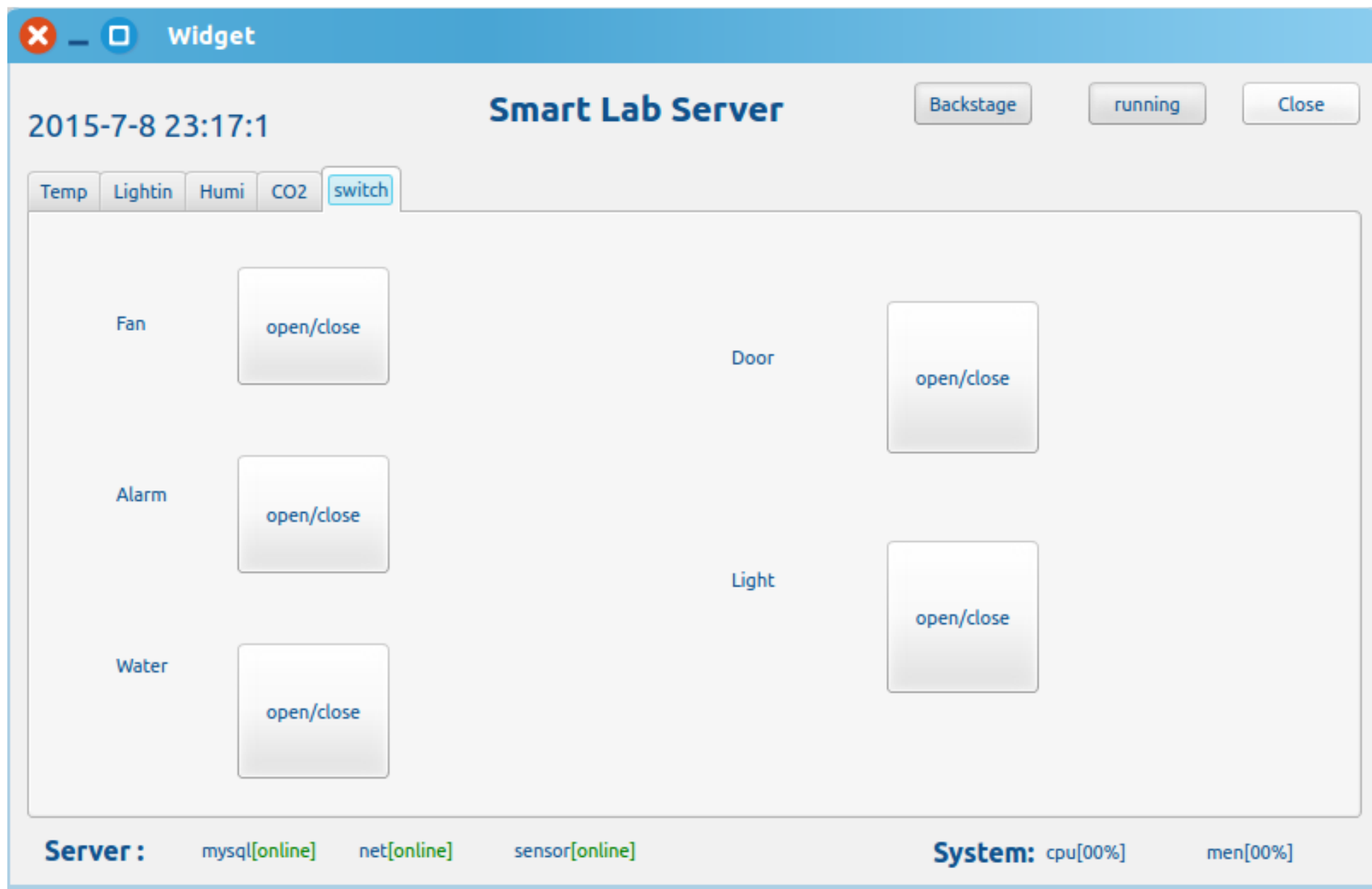
实验室环境监控主要分为三个部分，

一是实验室温度，湿度，光强等这些环境的状况参数。

二是实验设备的参数，例如设备的通断状态，功率，用电量等。

三是学生的信息。实验室监控子系统运行流程是：无线传感器检测实验室各相关参数，然后以无线通信的方式将监控数据发送无线网关，网关将数据打包后由基于**S5PV210**平台的**WEB**服务器及相关程序处理。监控数据主要包括室内温湿度检测、烟雾检测、有害气体检测和智能开关等。





实验室监控系统程序界面

×

□

Widget

2015-7-8 23:16:31

Smart Lab Server

Backstage

running

Close

Please login you Admin acount

username

6666

password

8888

Cancel

Login

1

2

3

back

4

5

6

0

7

8

9

clear

Server :

mysql[online]

net[online]

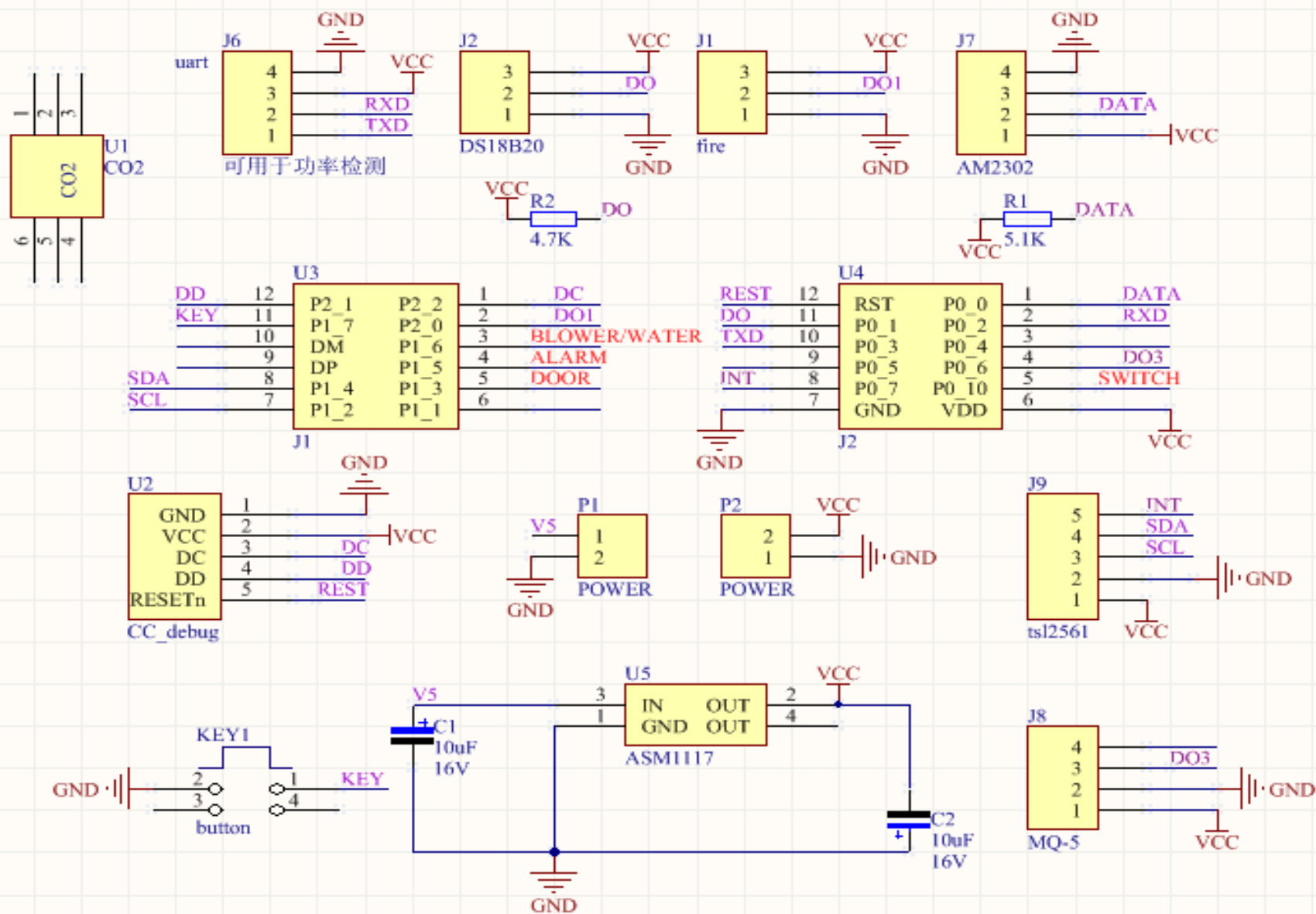
sensor[online]

System:

cpu[00%]

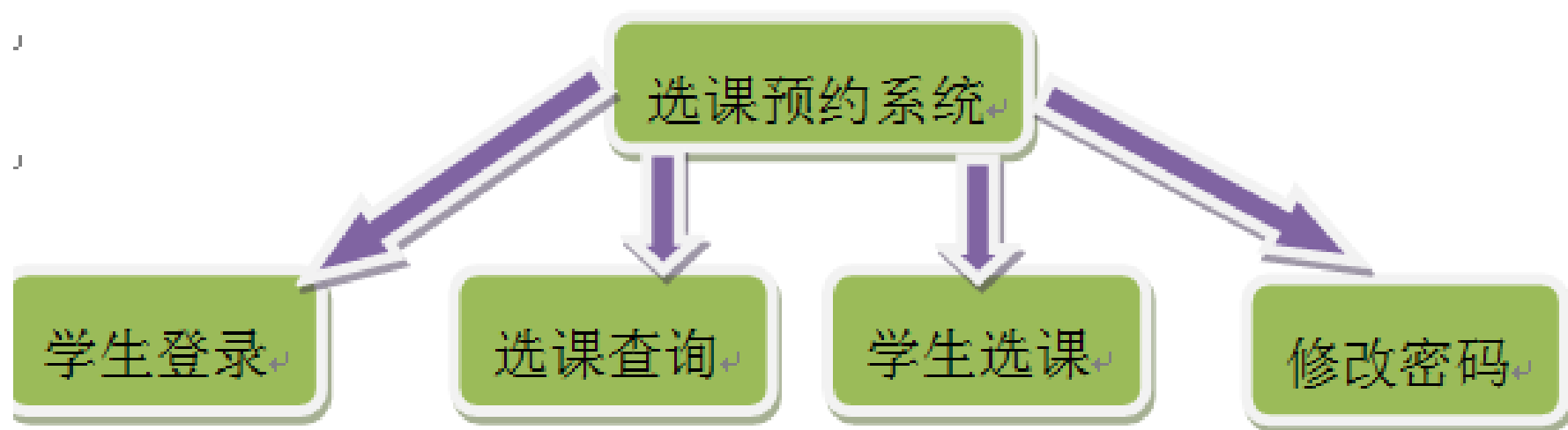
men[00%]

实验室监控系统节点控制界面

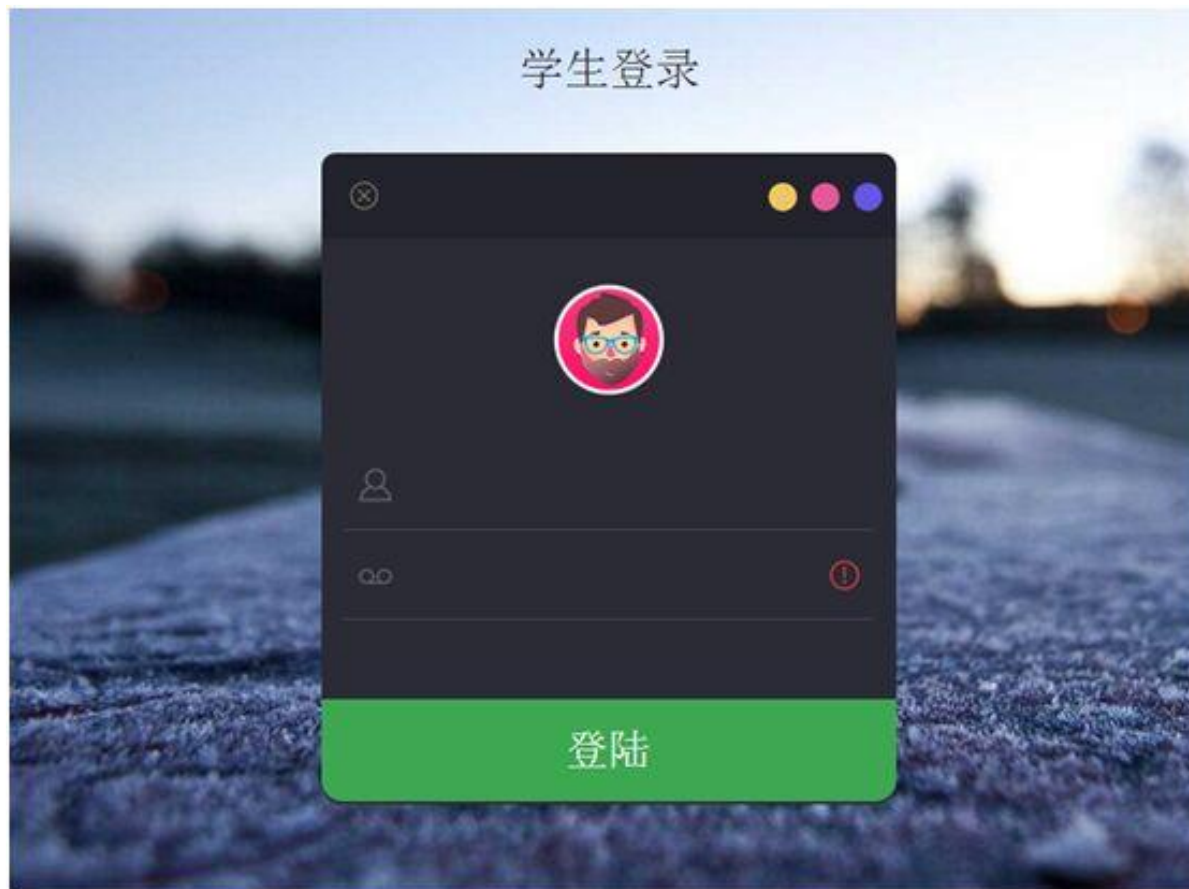


zigbee终端节点电路原理图

## 11.4.2 学生选课预约



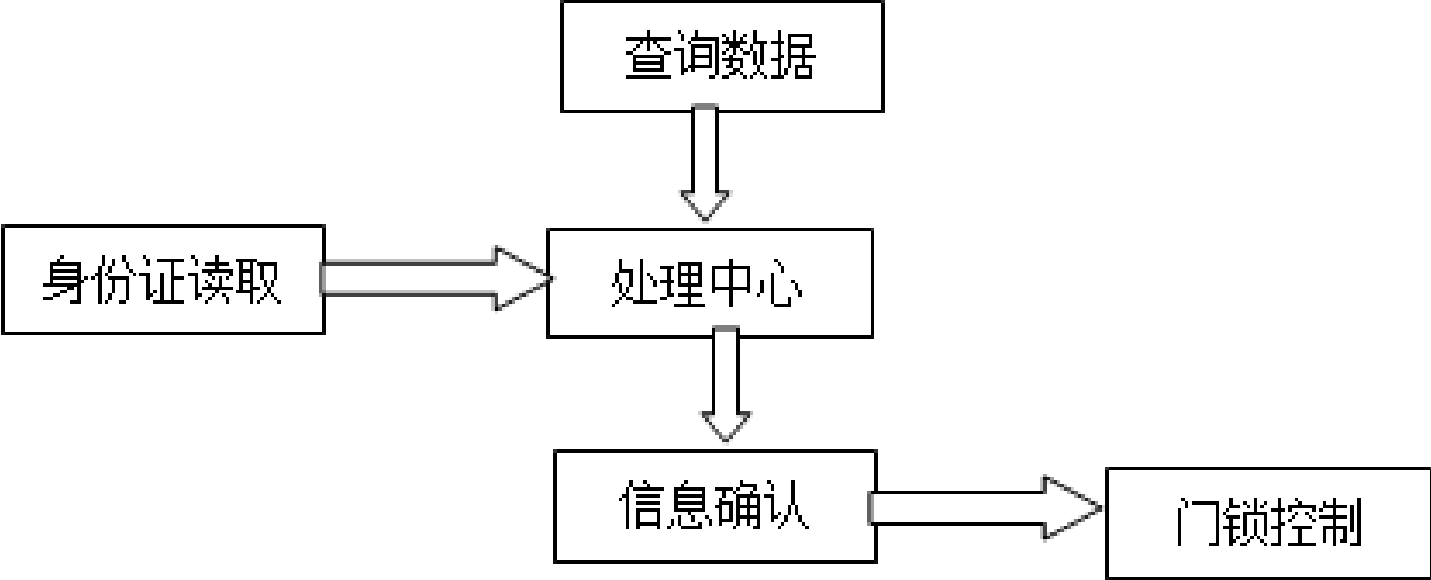
选课预约系统框图



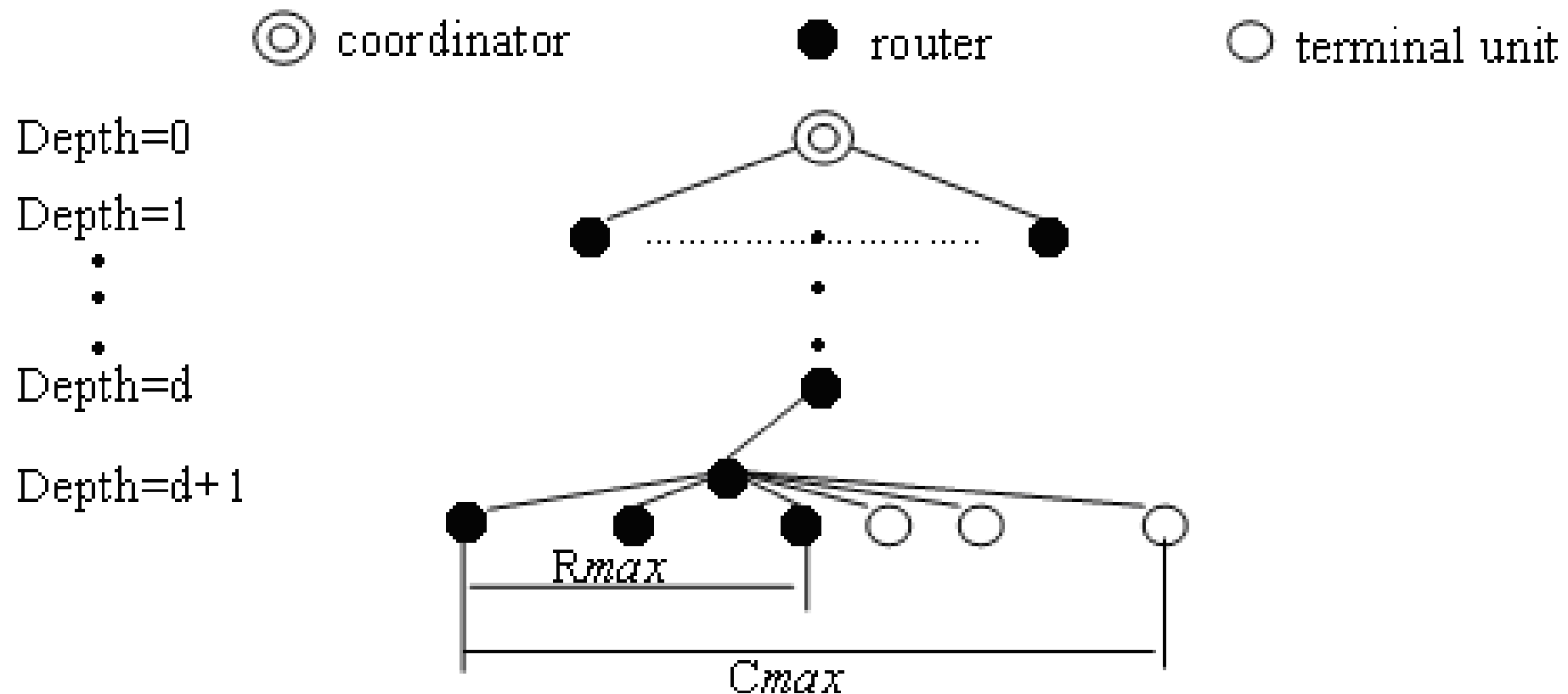
登录界面

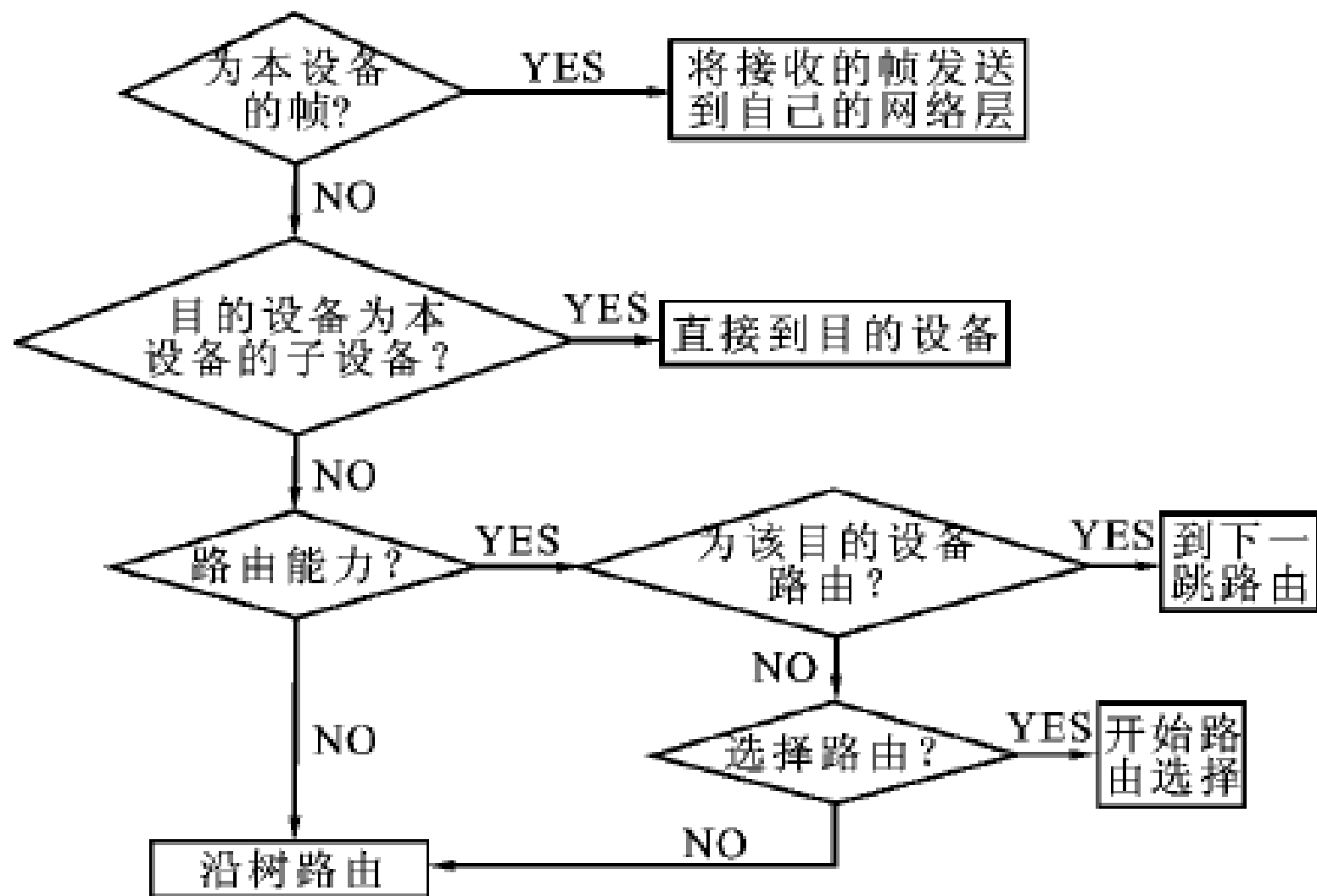
### 11.4.3 门禁系统

门禁系统使用了身份证读卡器结合S5PV210平台完成系统功能，学生进入实验室前必须先刷身份证以验证其身份，并在数据库中保存其相关信息，同时S5PV210平台作为控制中心将进行数据的匹配，当匹配成功时为该学生开门，并为该学生分配自己的实验仪器。



## 11.4.4 Zigbee网络的网络拓扑及路由协议





树路由算法



上位机上运行着由Qt开发的图形界面程序，该程序的主要功能是通过串口获取下位机网关的信息，并对信息进行解析，最终以图形的方式显示出整个zigbee网络节点的拓扑图。在这个拓扑图中每个节点所处的位置及节点传感器上的数据信息都可以实时显示出来

