

基于 REST 服务的最小物联网系统设计

目录

物联网毕业设计	4
绪论 ······	4
选题背景 ······	4
设计内容 ······	4
设计的目的及其意义 ······	5
国内外发展现状和趋势 ······	5
系统总体设计方案	5
硬件方案选择 ······	6
单片机选择 ······	6
软件方案选择 ······	7
数据通讯方式选择 ······	7
数据通信格式选择 ······	7
网络服务方案选择 ······	8
语言选择 ······	8
其它 ······	9
数据通讯设备 ······	9
辅助语言选择 ······	9
串口通信模块 ······	10
网页通信 ······	10
数据可视化框架选择 ······	10
本地系统设计	11
硬件设计 ······	11
Raspberry Pi ······	11
软件设计 ······	11
Arduino ······	12

Raspberry Pi	15
获取数据	16
串口通讯	17
python 串口通讯	17
网络系统设计	18
网络服务程序设计	18
基本的 REST 服务	20
系统前台设计	23
Ajax	23
系统后台设计	26

物联网毕业设计

绪论

选题背景

随着科技的发展，计算机电子技术迅猛发展，已经成为生活中不可缺少的部分。目前人们绝大多数都是采用 PC 进行网络数据传送，但由于成本高，限制了应用的范围。而嵌入式系统却越来越受到人们的青睐。它采用嵌入式的微处理器，支持 TCP/IP 协议，它已成为网络发展新阶段的标志。

物联网是新一代信息技术的重要组成部分。其英文名称是 **The Internet of things**。顾名思义，物联网的意思就是物物相连的互联网。这有两层意思：第一，物联网是建立在互联网之上的，是互联网的拓展和延伸；第二，其用户端扩展和延伸到了物品与物品之间，进行信息通信和交换。物联网有如下特征：

首先，广泛运用了各种感知技术。在物联网中部署了大量的多种传感器，每个传感器都能从外界采集信息，不同类的传感器捕获的信息不同。而且获得的数据具有实时性，按照一定的规律来采集数据，不断更新数据。

其次，它是建立在互联网上的网络。物联网技术的核心和基础仍是互联网，通过各种无线和有线网络与互联网结合起来，将物体的信息准确实时地传递出去，数据传输过程中必须适应各种网络协议。

还有，物联网本身也具有一种智能处理的能力，能够智能控制物体。物联网从传感器中获得数据，然后进行分析，处理处有意义的数据，来适应不同用户的需求。

设计内容

设计主要是关于基于 RESTful 服务的网络服务构建，可采用有线网络、无线网络、手机 GSM 网络等与 Internet 相关，通过手机、电脑、移动设备等登录到网页可实现控制家电的上的，并可实时查看诸如温度等一些信息的基本内容。

硬件设计时，采用 Arduino 单片机系统，作为一个基于 Atmega328 芯片的最小系统，Arduino 可以运行系统代码。Arduino 主要用于展示 LED 灯的控制，通过与 Raspberry PI 开发板相连来获取实时状态。Raspberry PI 作为一个 ARM 开发板，由于其运行的是 Linux 系统，在软件方面有着相对于其他开发板较好的支持，在这里是作为数据传输设备以用来进行模块分离。

软件设计时，由于一个物联网系统其核心是以网络为基础的，需要优先考虑网络方面的优化，也需要考虑数据库等的问题。

用户界面设计时，随着近来来平板、手机等移动设备的流行，在设计时不能再以桌面程序为核心，需要考虑不同设备之间的兼容性等问题，这里便以网页为核心作为显示。而，随着云计算技术的流行，未来的物联网系统必然也会基于云计算技术构建。作为一个可视化的网页来说，实时的状态显示等是较为重要，同时我们需要考虑的是用户体验。

设计的目的及其意义

设计以简化物联网系统为主，简化一个可扩展的最小的物联网系统，以简化系统的逻辑为起点，为广大的用户提供一个良好的了解物联网系统方面知识的框架。

国内外发展现状和趋势

物联网是建立在互联网技术之上的。目前，我国物联网发展与全球同处于起步阶段，初步具备了一定的技术、产业和应用基础，呈现出良好的发展态势。把单片机应用系统和 Internet 连接也是一种趋势。

目前无线通信网络已经覆盖各地，是实现 ``物联网" 必不可少的设施，可以将安置在每个物品上的电子介质产生的数字信号通过无线网络传输出去。``云计算" 技术的运用，使数以亿计的各类物品的实时动态管理变得可能。

物联网技术的推广已经取得一定的成效。在多方面已经开始应用，如远程抄表，电力行业，视频监控等等。以及在物流领域和医疗领域也都日趋成熟，如物品存储及运输监测，远程医疗，个人的健康监护等。除此之外在环境监控，楼宇节能，食品等方面也开展了广泛应用。

尽管在这些领域已经取得一些进展，但应认识到，物联网发展技术还存在一系列制约和瓶颈。有几个方面可以表现出来：核心技术与国外差距较大，集成服务能力不够，缺乏骨干企业，应用水平不高，信息安全存在隐患。我们国家在 PC 架构领域还没有主动权，软件产品很少。目前，计算环境正在向以网络为中心发展，有很多产品不必也 windows 兼容，因此，研究单片机系统接入网络，前途宽广。

系统总体设计方案

物联网的核心也就是网络服务，而网络服务在某种意义上来说，就是需要打造一个跨平台的通信协议，在使机器、家电、设备等连上计算机网络。基本的物联网系统，不仅能控制设备，还可以在远程查看状态。而复杂的物联网系统可以让互联网上的设备之间实现互联与通信，也就是物联网的最终目标所在 -----使物体与物体之间的交互成为可能，不需要人为去干预。

设备在现实世界就是一种资源，在互联网上也应该是一种资源，互联网上的网页就相当于是一种资源。

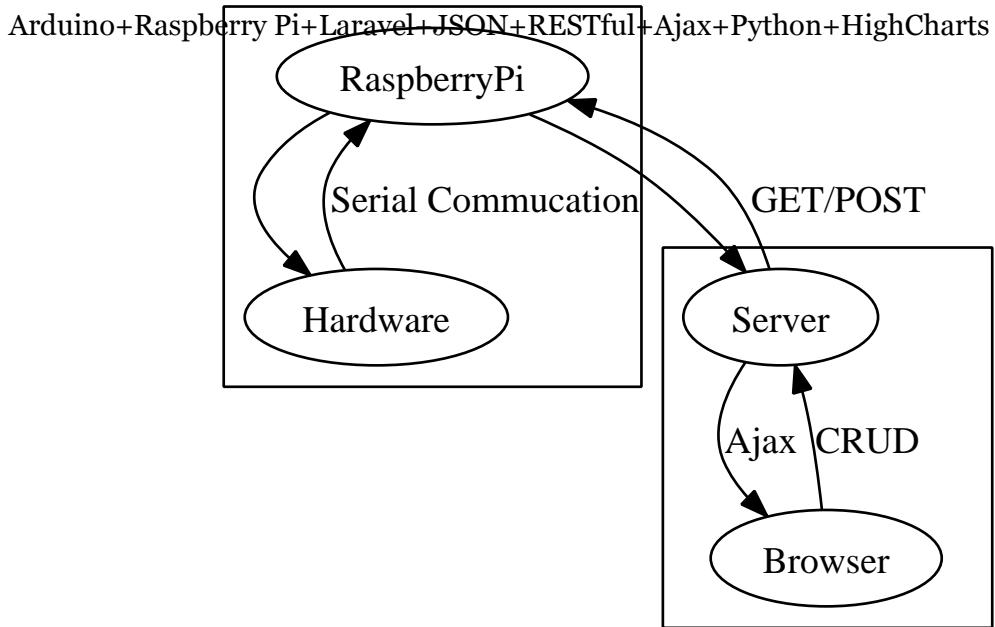


图 1: 系统框架图

Arduino 与 Raspberry Pi 通过串口通信的方式实现通信，相互传输所需要的数据，Raspberry Pi 将资源传于互联网上对应的接口，接口可以在互联网上被访问。Laravel 框架架构架于服务器之上，将 Raspberry Pi 获取过来的数据存储于 MySQL 数据，再以 REST 服务的方式共享数据，互联网上的其他设备便可以通过网络来访问这些设备。Ajax 用于将后台的数据以不需要刷新的方式传递到网站前台，通过 HighCharts 框架显示给终端用户。

硬件方案选择

单片机选择

Arduino Arduino，是一个开放源代码的单芯片微电脑，它使用了 Atmel AVR 单片机，采用了基于开放源代码的软硬件平台，构建于开放源代码 simple I/O 接口板，并且具有使用类似 Java, C 语言的 Processing/Wiring 开发环境。

Arduino 开发板封装了常用的库到开发环境中，可以让用户在开发产品时，将主要注意力放置于所需要实现的功能上，而不是开发的过程中。在为 Arduino 写串口程序

时，我们只需要用 `Serial.begin(9600)` 以 9600 的速率初始化串口，而在往串口发送数据时，可以用 `Serial.write('1')` 的方式向串口发送字串'1'。

51 单片机¹，又称微控制器，是把中央处理器、存储器、定时/计数器 (Timer/Counter)、各种输入输出接口等都集成在一块集成电路芯片上的微型计算机。与应用在个人计算机中的通用型微处理器相比，它更强调自供应（不用外接硬件）和节约成本。它的最大优点是体积小，可放在仪表内部，但存储量小，输入输出接口简单，功能较低。

51 单片机相较于 Arduino 开发板，不仅代码复杂，由于系统比较古老而不方便于快速开发。

软件方案选择

数据通讯方式选择

REST REST² 从资源的角度来观察整个网络，分布在各处的资源由 URI 确定，而客户端的应用通过 URI 来获取资源的表征。获得这些表征致使这些应用程序转变了其状态。随着不断获取资源的表征，客户端应用不断地在转变着其状态，所谓表征状态转移。

SOAP 简单对象访问协议是交换数据的一种协议规范，使用在计算机网络 Web 服务中，交换带结构信息。SOAP 为了简化网页服务器从 XML 数据库中提取数据时，节省去格式化页面时间，以及不同应用程序之间按照 HTTP 通信协议，遵从 XML 格式执行资料互换，使其抽象于语言实现、平台和硬件。

数据通信格式选择

JSON JSON³是一种轻量级的数据交换格式。易于人阅读和编写。同时也易于机器解析和生成。它基于 JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999 的一个子集。JSON 采用完全独立于语言的文本格式，但是也使用了类似于 C 语言家族的习惯⁴。这些特性使 JSON 成为理想的数据交换语言。

JSON 相对于 XML 来说可以减少文件的大小，同时我们可以用于网站前端的数据通讯。

对于基于浏览器的客户端使用的 web 服务更倾向于使用 JSON 作为表述格式。

¹全称单片微型计算机（英语：Single-Chip Microcomputer）

²Representational State Transfer

³JavaScript Object Notation

⁴包括 C, C++, C#, Java, JavaScript, Perl, Python 等

XML 可扩展标记语言⁵，是一种标记语言。标记指计算机所能理解的信息符号，通过此种标记，计算机之间可以处理包含各种信息的文章等。如何定义这些标记，既可以选择国际通用的标记语言，比如 **HTML**，也可以使用像 **XML** 这样由相关人士自由决定的标记语言，这就是语言的可扩展性。**XML** 是从标准通用标记语言（**SGML**）中简化修改出来的。它主要用到的有可扩展标记语言、可扩展样式语言（**XSL**）、**XBRL** 和 **XPath** 等。

XML 具有良好的可读性，有着较好的库支持，从 **Java** 语言到其他语言，如 **Linux** 系统上 **libxml** 等对 **XML** 的支持比较好。

网络服务方案选择

语言选择

PHP Laravel

PHP⁶ 是一种开源的通用计算机脚本语言，尤其适用于网络开发并可嵌入 **HTML** 中使用。**PHP** 的语法借鉴吸收了 **C** 语言、**Java** 和 **Perl** 等流行计算机语言的特点，易于一般程序员学习。**PHP** 的主要目标是允许网络开发人员快速编写动态页面，但 **PHP** 也被用于其他很多领域。

Laravel

Laravel 是一套简洁、优雅的 **PHP Web** 开发框架。它可以让你从面条一样杂乱的代码中解脱出来；它可以帮你构建一个完美的网络 **APP**，而且每行代码都可以简洁、富于表达力。

Java Spring

Java

Java 是一种可以撰写跨平台应用软件的面向对象的程序设计语言，是由 **Sun Microsystems** 公司于 1995 年 5 月推出的 **Java** 程序设计语言。**Java** 技术具有卓越的通用性、高效性、平台移植性和安全性，广泛应用于个人 **PC**、数据中心、游戏控制台、科学超级计算机、移动电话和互联网，同时拥有全球最大的开发者专业社群。在全球云计算和移动互联网的产业环境下，**Java** 更具备了显著优势和广阔前景。

Spring

Spring 是一个开源框架，是为了解决企业应用程序开发复杂性。**Spring** 框架的主要优势之一就是其分层架构，分层架构允许使用者选择使用哪一个组件，同时为 **J2EE** 应用程序开发提供集成的框架。**Spring** 使用基本的 **JavaBean** 来完成以前只可能由 **EJB** 完成的工作。

⁵eXtensible Markup Language，简称：**XML**

⁶PHP：Hypertext Preprocessor，即超文本预处理器

成的事情。然而，Spring 的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何 Java 应用都可以从 Spring 中受益。

由于相较于 Java 在 web 方面没有 PHP 来得快速、简单、有效，同时 Laravel 框架在某些方面如数据迁移、代码生成比 Spring 快，同时不需要依赖于开发环境，这里以 Laravel 作为框架，可以利用 artisan 工具等的强大驱动开发。

其它

数据通讯设备

Raspberry PI

Raspberry Pi 是一款针对电脑业余爱好者、教师、小学生以及小型企业等用户的迷你电脑，预装 Linux 系统，体积仅信用卡大小，搭载 ARM 架构处理器，运算性能和智能手机相仿。在接口方面，Raspberry Pi 提供了可供键鼠使用的 USB 接口，此外还有千兆以太网接口、SD 卡扩展接口以及 1 个 HDMI 高清视频输出接口，可与显示器或者 TV 相连。

Linux 是一套免费使用和自由传播的类 Unix 操作系统，是一个基于 POSIX 和 UNIX 的多用户、多任务、支持多线程和多 CPU 的操作系统。它能运行主要的 UNIX 工具软件、应用程序和网络协议。它支持 32 位和 64 位硬件。Linux 继承了 Unix 以网络为核心的设计思想，是一个性能稳定的多用户网络操作系统。

Raspberry Pi 相比于一般的 ARM 开发板来说，由于其本身搭载着 Linux 操作系统，可以用诸如 Python、Ruby 或 Bash 来执行脚本，而不是通过编译程序来运行，具有更高的开发效率。

辅助语言选择

Python

Python，是一种面向对象、直译式计算机程序设计语言，由 Guido van Rossum 于 1989 年底发明，第一个公开发行版发行于 1991 年。Python 语法简洁而清晰，具有丰富和强大的类库。它常被昵称为胶水语言，它能够很轻松的把用其他语言制作的各种模块（尤其是 C/C++）轻松地联结在一起。常见的一种应用情形是，使用 python 快速生成程序的原型（有时甚至是程序的最终界面），然后对其中有特别要求的部分，用更合适的语言改写，比如 3D 游戏中的图形渲染模块，速度要求非常高，就可以用 C++ 重写。

Ruby

Ruby, 一种为简单快捷的面向对象编程（面向对象程序设计）而创的脚本语言，在 20 世纪 90 年代由日本人松本行弘开发，遵守 GPL 协议和 Ruby License。

Python 相对于 **Ruby** 有着更好的跨平台能力，同时有良好的可读性，加之 **Ruby** 语言没有对串口通讯及 **Windows** 系统更好的支持。

串口通信模块

PySerial

PySerial 封装了串口通讯模块，支持 Linux、Windows、BSD(可能支持所有支持 POSIX 的操作系统)，支持 Jython(Java) 和 IronPython(.NET and Mono).

在使用 **PySerial** 之后，我们只需要

```
ser=serial.Serial("/dev/ttyACM0", 9600)
ser.write("1")
```

就可以向串口发送一个字符 1。

网页通信

Ajax **AJAX**⁷ 是由 Jesse James Gaiett 创造的名词，是指一种创建交互式网页应用的网页开发技术。

系统主要用 **Ajax** 来实现实时温度显示，通过直接访问 **JSON** 数据的情况下，可以在不需要刷新页面的情况下直接读取数据。

数据可视化框架选择

HighCharts

Highcharts 是一个用纯 **JavaScript** 编写的一个图表库，能够很简单便捷的在 **web** 网站或是 **web** 应用程序添加有交互性的图表，并且免费提供给个人学习、个人网站和非商业用途使用。**HighCharts** 支持的图表类型有曲线图、区域图、柱状图、饼状图、散状点图和综合图表。

D3.js

⁷Asynchronous JavaScript and XML (异步 JavaScript 和 XML)

本地系统设计

硬件设计

Raspberry Pi

Raspberry Pi 开发板在这里主要工作有:

- 与 Arduino 开发板，通过 USB 线连接。
- 可以直接运行 Debian GNU/Linux 系统，通过网线上网，并从服务器中读取数据，
- 通过 Python 语言接收、发送串口数据。

软件设计

在本地我们需要解决的问题可以如下描述，Arduino 开发板从串口一直读取数据，Raspberry Pi 从 URL 中验证数据、解析数据，再将数据发送到串口，我们可以用下面的伪代码来描述：

```
arduino:
    begin
        repeat
            wait(serial.open)
            data:=receive_data()
            led_status:=parse(data)
            if led_status
                oped(led.id)
            util false
    end

raspberrypi:
    begin
        repeat
            json:=get_data(url)
            if validate(json).success()
                data:=parse(json)
                serial.write(data)
            util false
    end
```

Arduino

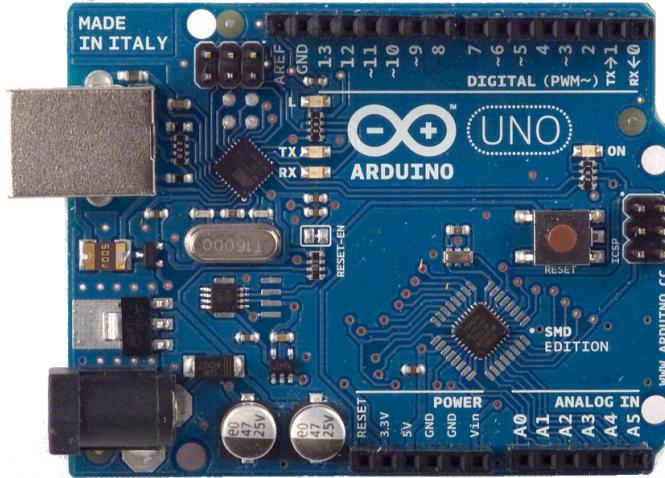


图 2: Arduino 开发板

Arduino UNO 用的处微控制器是 Atmega328，它与 Arduino 芯片的对应关系如下所示

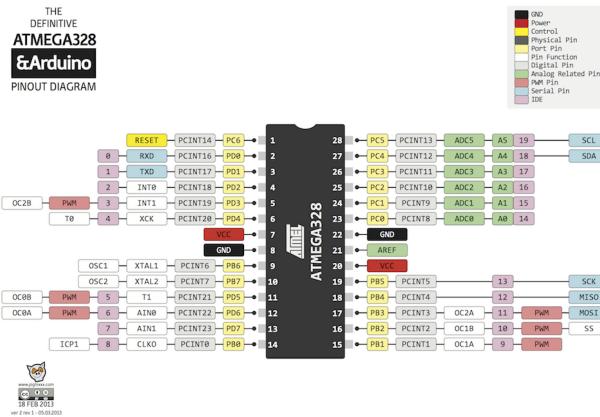


图 3: Arduino 管脚 Atmega328 对应图

其主要参数如下所示：

高性能, 低功耗的 AVR®8 位微控制器 • 先进的 RISC 结构

- 131 条指令
- 绝大多数为单时钟周期执行
- 32 个通用工作寄存器
- 全静态工作
- 高达 20 MIPS 的吞吐量, 在 20 MHz
- 片上 2 周期乘法器高耐用性非易失性内存段

- 8K 字节的系统内可编程 Flash 存储器 (ATMEGA88PA 中)
- 512 字节的 EEPROM (ATMEGA88PA)
- 1K 字节的片内 SRAM (ATMEGA88PA)
- 写/擦除次数: 10,000 次,000 EEPROM
- 数据保存: 20 年在 85°C/100 年在 25°C(1)
- 可选的引导具有独立锁定位代码段在系统编程的片上引导程序真正的同时读写操作
- 编程软件安全锁外设特点
- 两个 8 位定时器/计数器具有独立预分频器和比较模式
- 1 个 16 位定时器/计数器具有独立预分频器, 比较模式, 并捕获模式
- 具有独立振荡器的实时计数器
- 6 个 PWM 通道
- 8 通道 10 位 ADC 在 TQFP 和 QFN / MLF 封装温度测量
- 6 通道 10 位 ADC 引脚 PDIP 封装温度测量
- 可编程的串行 USART
- 主/从机模式的 SPI 串行接口
- 面向字节的两线串行接口 (飞利浦公司的 I2C 兼容)
- 独立的片内振荡器的可编程看门狗定时器
- 片上模拟比较器
- 中断和唤醒引脚电平变化单片机的特殊功能
- 上电复位以及可编程的掉电检测
- 内部校准振荡器
- 外部和内部中断源
- 6 种睡眠模式: 空闲模式,ADC 噪声抑制, 省电, 掉电, 待机, 扩展 Standby

Arduino 部分硬件程序如下所示, 主要负责从串口中读入数据, 并用 led 灯显示。程序流程图如下所示

系统主要的功能在于接收和传递数据。

代码如下所示

```
void setup() {  
    Serial.begin(9600);  
    pinMode(13, OUTPUT);  
    pinMode(12, OUTPUT);
```

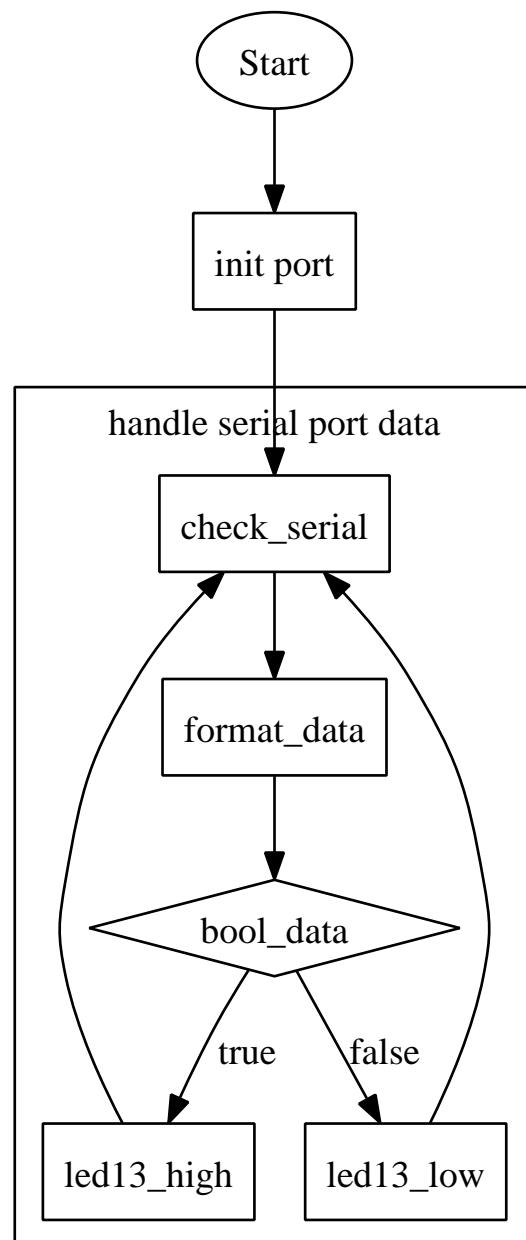


图 4: Arduino 程序流程图

```

    }

    int serialData;
    void loop() {
        String inString = "";
        while (Serial.available() > 0)
        {
            int inChar = Serial.read();
            if (isDigit(inChar)) {
                inString += (char)inChar;
            }
            serialData = inString.toInt();
            Serial.print(serialData);
        }
        if (serialData == 1) {
            digitalWrite(12, LOW);
            digitalWrite(13, HIGH);
        } else {
            digitalWrite(13, LOW);
            digitalWrite(12, HIGH);
        }
    }
}

```

Raspberry Pi

Raspberrypi 如下所示的开发板

```

begin
repeat
    json := get_data(url)
    if validate(json).success
        data := parse(json)
        if data == 1
            serial_send("1")
        else
            serial_send("0")
    else

```



图 5: Raspberry Pi 开发板

```
        output "error"  
        util false  
    end
```

Raspberry Pi 程序

图 6: Python 程序流程图

获取数据

Raspberry Pi 端的主要功能便是将数据从 <http://www.xianuniversity.com/athome/> [^domain] 下载下来并解析数据，再将数据用串口通讯的方式传递给 Arduino。

在 Debian 系统中，自带了 python 语言，python 有良好的动态特性，同时有强大的自建库功能。在 python 语言中可以用自带的 `urllib2` 库打开并下载网页的内容，将上述网址中的 JSON 数据下载到本地。

数据采用的是 JSON 格式，具有良好的可读性，同时方便于解析，相比于 XML 格

式又可以减少文件大小，

```
[ {
    "id": 1,
    "temperature": 10,
    "sensors1": 22,
    "sensors2": 11,
    "led1": 0
}]
```

JSON 的将上述中的数据取出来后，通过 python 中的 json 库，将 json 数据转换为数组，将取出数据中的第一个结果中的 id 的值。

串口通讯

由于 python 中没有用于串口通讯的库，需要寻找并安装这样一个库，这里就用到了 pip 这样的包管理工具 -----用于管理 python 的库。

安装 **pyserial** pip 常用命令有 install、uninstall 以及 search，install 顾名思义就是安装，安装 pip 库如下所示⁸，如后代码如下所示，\$⁹开头：

```
$pip install pyserial
```

python 串口通讯

在 Linux 内核的系统¹⁰中虚拟串口用的节点是 ttyACM，位于/dev 目录下。

```
serial.Serial("/dev/ttyACM0", 9600)

8在 Windows 系统中需要先安装 pip，再安装 pyserial。
9指在 *nix 系统的终端中执行的命令。
10在 Windows 系统上，只需要将/dev/ttyACM0 改为对应的 com 口。
import json import urllib2 import serial import time
url='`http://www.xianuniversity.com/athome/1'
while 1: try: date=urllib2.urlopen(url) result=json.load(date) status=result[0]['`led1']
ser=serial.Serial(``/dev/ttyACM0",9600) if status==1 : ser.write(``1") elif status==0: ser.write(``0")
time.sleep(1) except urllib2.URLError: print ``Bad URL or timeout"
```

串行接口是一种可以将接受来自 CPU 的并行数据字符转换为连续的串行数据流发送出去，同时可将接受的串行数据流转换为并行的数据字符供给 CPU 的器件。一般完成这种功能的电路，我们称为串行接口电路。

便是打开这个设备，以 9600 的速率传输数据。

```
fdhuang@phodal:python (master)$ python get.py
[{"led1": 1, "sensors2": 12, "id": 1, "sensors1": 18, "temperature": 14}]
[{"led1": 1, "sensors2": 12, "id": 1, "sensors1": 18, "temperature": 14}]
[{"led1": 1, "sensors2": 12, "id": 1, "sensors1": 18, "temperature": 14}]
[{"led1": 1, "sensors2": 12, "id": 1, "sensors1": 18, "temperature": 14}]
[{"led1": 1, "sensors2": 12, "id": 1, "sensors1": 18, "temperature": 14}]
[{"led1": 1, "sensors2": 12, "id": 1, "sensors1": 18, "temperature": 14}]
[{"led1": 1, "sensors2": 12, "id": 1, "sensors1": 18, "temperature": 14}]
```

图 7: python 返回 json 数据

系统还需要对上面的数据进行处理，只拿其中的结果。

```
fdhuang@phodal:python (master)$ python get.py
[{"temperature": 14}]

0
0
0
0
0
0
0

```
 网站采用的是JSON格式，具有良好的可读性，同时方便于解析，相比
 又可以减少文件大小。
```
```
 javascript
```

图 8: python 处理完后的结果

当改变 led 的状态后，便可以得到下面的结果

## 网络系统设计

### 网络服务程序设计

对于物联网系统网络的核心是构建一个 RESTful 服务，而这构建 RESTful 的核心便是基础的 HTPP 协议。基础的 HTTP 协议便是:GET、POST、PUT、DELETE。它们分别对应四种基本操作：GET 用来获取资源，POST 用来新建资源（也可以用于更新资源），PUT 用来更新资源，DELETE 用来删除资源。

```
fdhuang@phodal:python (master)$ python get.py
0
0
0
0
1
1
1
1
1
1
1
1
```

图 9: 改变状态后的结果

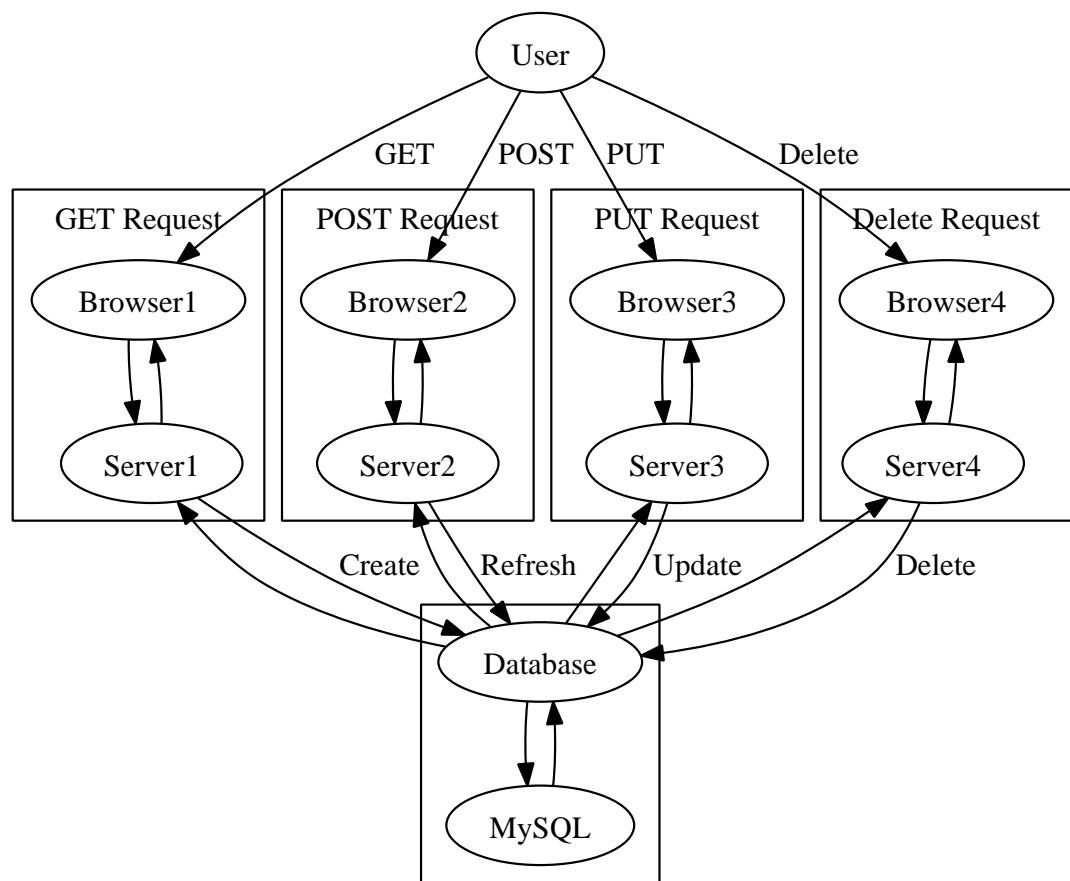


图 10: restful

简要的来说，一个 **GET** 动作便是在打开一个网页的时候，看到的内容，便是 **GET** 到的资源。而在获取取到网页的内容之前，会有一个 **POST** 动作到所要打开的网站的服务器。

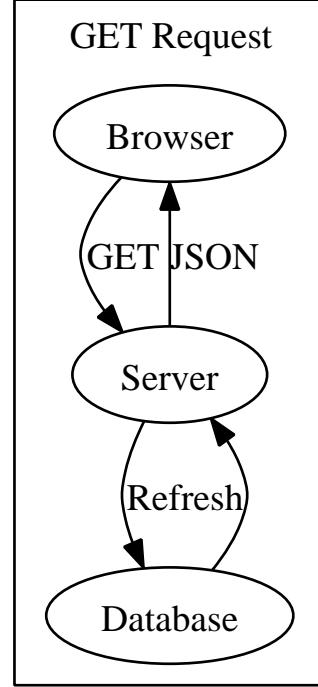


图 11: Get Request

### 基本的 **REST** 服务

**REST** 服务实际上是充当着网络与设备的传输介质，构建一个 **REST** 服务也就相当于获取一个 URL 下的某个数据

```
$curl http://www.xianuniversity.com/athome/1
```

返回结果如下所示

假设有这样一个资源用于呈现 led 的状态，即 <http://localhost/status/1><sup>11</sup>，获取这个 LED 的状态便发出了类似下面这样的请求：

```
GET /status/1 HTTP/1.1
Host:localhost
Content-Type:application/json;charset=UTF-8
```

---

<sup>11</sup>在本地进行 web 开发时，浏览器可以识别 localhost，配置好 Hosts 时相当于 127.0.0.1。

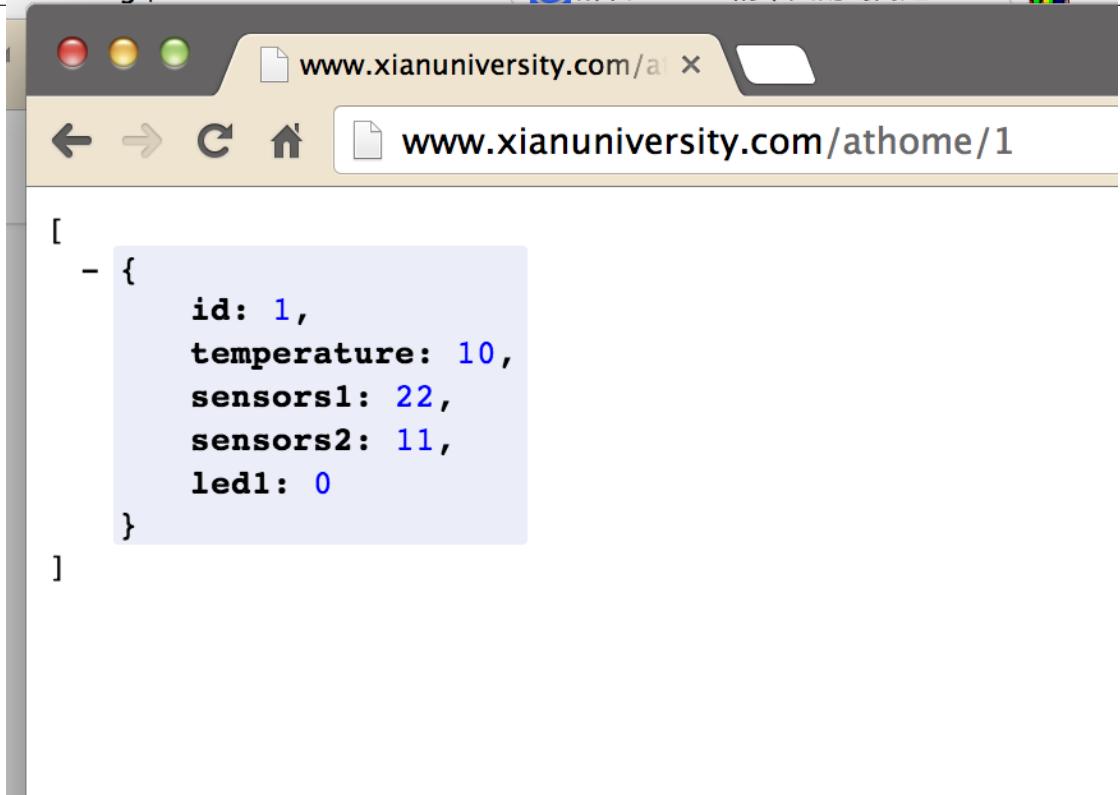


图 12: JSON 结果返回图

在客户端发出上述的请求的时候，服务端需要对其做出响应，构造出一个下面的结果

```
[{
 "status": 1
}]
```

1 代表给予灯的状态应该是亮的，在那之后需要做的便是将其通过串口发送给单片机进行处理，对应于一个关机的结果便是

```
[{
 "status": 0
}]
```

这样就完成了基本的状态设计。而对于系统最后需要解析的数据的结果来说，还需要加入其他元素，

```
[{
 "id": 1,
 "status": 1
}]
```

```

 "temperature": 10,
 "sensors1": 22,
 "sensors2": 11,
 "led1": 0
}

```

这里也涉及到了 json 数据结构的设计，可以将上面的结果设计为

```

[{
 "id": 1,
 "temperature": 10,
 "sensors": [
 {
 "sensor": 22,
 "sensor": 11,
 },
 "led1": 0
}
]

```

这种具有更好的可读性，然而在对于网速速度要求高的情况下，会表现得不好，同时会造成额外的系统开销。对于这样一个需要不断读取数据的系统来说，采用单层结构的 json 数据会更具有优势。

在设计这样一个接口的时候，需要考虑客户端可能需要获取全部的数据

```

GET /status HTTP/1.1
Host:localhost
Content-Type:application/json;charset=UTF-8

```

设计好这样的接口有助于显示在系统的前台，而这也是无法在物联网系统中产生统一协议的原因之一，复杂的接口无法用于简单功能的场景。

下面是一个简单的 POST 请求的示例，系统需要能接收 POST 请求，并将请求存储到数据库

```

POST / HTTP/1.1
Host:localhost
User-Agent: Go 1.1 package http
Content-Length: 45

```

**Authorization:** 123456

**Accept-Encoding:** gzip

一个 **PUT** 动作但是我们更新资源，就好比是我们创建一个日志或者一个说说一样。**DELETE** 动作，便是删除动作了，而这也是一个物联网系统服务所需要的。

## 系统前台设计

在对系统前台设计的时候，在考虑不同移动设备的兼容的同时，也需要保持一个良好可用的结构。而系统在前台的主要功能是在于控制物体的状态、显示一些数值的变化，控制物体状态的关键在于如何将数据由前台 **POST** 到后台，在网页端可以用 **POST**，而在移动端则可以用 **JSON API**。

### Ajax

- **AJAX : Asynchronous JavaScript and XML** (异步的 JavaScript 和 XML)。
- **AJAX** 不是新的编程语言，而是一种使用现有标准的新方法。
- **AJAX** 是与服务器交换数据并更新部分网页的艺术，在不重新加载整个页面的情况下。

剥离后的 **Ajax** 部分代码如下所示，主要用的是 **jQuery** 框架的 **getJSON** 来实现的

```
begin
 data:=get_data(url)
 if data.get_success
 temperature:=data.push(temperature)
```

当按下 **Change Status** 按钮时，系统发生了如下变化

系统会先向服务器发送数据，也就是 **POST** 请求，在请求结束后，系统将会刷新页面，也就是 **GET** 请求。

系统会不断从后台获取数据结果，如下所示

在 **Javascript** 语言中有函数库可以直接用于获取后台数据 -----**getJSON**，可以从指定的 **URL** 中获取结果。

- **url** 用于提供 **json** 数据的地址页
- **data(Optional)** 用于传送到服务器的键值对

The screenshot shows a web application window with the URL [p.pnoodai.com/athome/1/edit](http://p.pnoodai.com/athome/1/edit) in the address bar. The main content area is titled "Edit 1". It contains three input fields: "开关1" (Switch 1) with the value "开" (On), "传感器1" (Sensor 1) with the value "18", and "传感器2" (Sensor 2) with the value "12". Below these is a "温度传感器" (Temperature Sensor) field with the value "14". At the bottom is a blue button labeled "Change Status!". The footer of the page includes a copyright notice: "© Company 2013".

图 13: 控制界面

The screenshot shows a log viewer with the following interface elements: a toolbar with icons for search, filter, and preserve log; a table header with columns for Name, Path, Method, and Status Text; and two log entries.

| Name | Path      | Method | Status Text |
|------|-----------|--------|-------------|
|      | 1 /athome | POST   | 302 Found   |
|      | athome    | GET    | 200 OK      |

图 14: GET POST 数据

系统前台设计

Preserve log

网络系统设计

| Name<br>Path  | Method | Status<br>Text       |
|---------------|--------|----------------------|
| 5<br>/athome  | GET    | 200<br>OK            |
| 5<br>/athome  | GET    | 200<br>OK            |
| athome/       | GET    | 301<br>Moved Permane |
| <u>athome</u> | GET    | 200<br>OK            |
| 5<br>/athome  | GET    | 200<br>OK            |
| 5<br>/athome  | GET    | 200<br>OK            |
| athome/       | GET    | 301<br>Moved Permane |
| athome        | GET    | 200<br>OK            |

图 15: 后台获取数据

- callback(Optional) 回调函数，json 数据请求成功后的处理函数

```

var dataLength = [];
function drawTemp() {
 var zero = [];
 $.getJSON('/athome/', function(json) {
 var items = [];
 dataLength.push(json.length);
 $.each(json, function(key, val) {
 zero.push(val.temperature);
 });
 });
}

```

实际上，我们做的只是从 /athome/ 下面获取数据，再将数据堆到数组里面，再把这部分放到图形中。

- 兼容性：兼容当今所有的浏览器，包括 iPhone、IE 和火狐等等；
- 对个人用户完全免费；

- 纯 JS, 无 BS;
- 支持大部分的图表类型：直线图，曲线图、区域图、区域曲线图、柱状图、饼装图、散布图；
- 跨语言：不管是 PHP、Asp.net 还是 Java 都可以使用，它只需要三个文件：一个是 Highcharts 的核心文件 highcharts.js, 还有 a canvas emulator for IE 和 Jquery 类库或者 MooTools 类库；
- 提示功能：鼠标移动到图表的某一点上有提示信息；
- 放大功能：选中图表部分放大，近距离观察图表；
- 易用性：无需要特殊的开发技能，只需要设置一下选项就可以制作适合自己的图表；
- 时间轴：可以精确到毫秒；

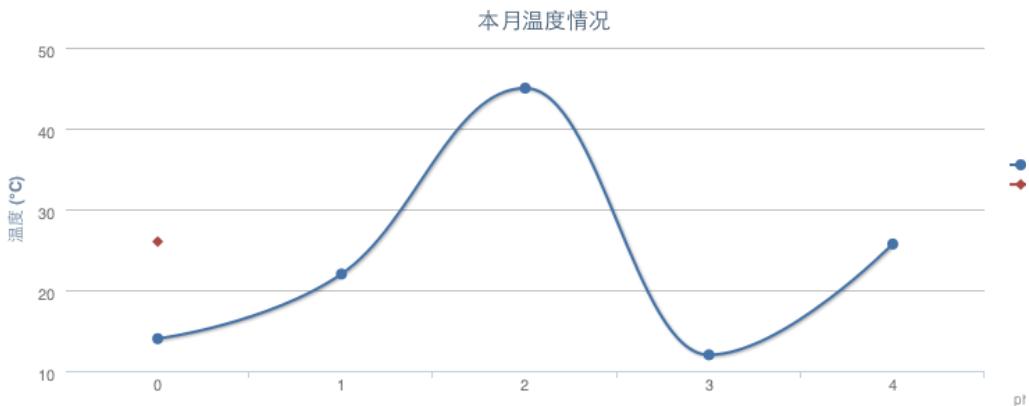


图 16: chart

## 系统后台设计

系统使用 Laravel 框架作为系统底层，需要配置其运行环境<sup>12</sup>，创建数据库<sup>13</sup>，对应于上面生成的最后的 JSON 数据，创建对应的数据库：

系统数据库结构如下所示

| 表名 | 数据类型 | 作用 |
|----|------|----|
| id | 整数   | ID |

<sup>12</sup>这里用的是 Linux+Nginx+MySQL+PHP。

<sup>13</sup> 创建数据的代码:CREATE DATABASE IF NOT EXISTS bbs default charset utf8 COLLATE utf8\_general\_ci;

|             |     |       |
|-------------|-----|-------|
| temperature | 浮点数 | 温度    |
| sensors1    | 浮点数 | 传感器1  |
| sensors2    | 浮点数 | 传感器2  |
| led1        | 布尔型 | led状态 |

+-----+-----+-----+

对应于数据库，编写相应的程序

```
public function up()
{
 Schema::create('athomes', function(Blueprint $table)
 {
 $table->increments('id');
 $table->float('temperature');
 $table->float('sensors1');
 $table->float('sensors2');
 $table->boolean('led1');
 $table->timestamps();
 });
}
```

当 POST 数据的时候，便是将数据存往数据库，而 GET 的时候则是从数据库中拿出数据再渲染给浏览器，GET、PUT、DELETE、POST 便是对就于数据库的 Create、Refresh、Update、Delete

系统使用 MySQL 作为数据库，开始的时候需要创建数据库，在数据库中执行

```
CREATE DATABASE IF NOT EXISTS iot default charset utf8 COLLATE utf8_general_ci
```

以创建 athomes 这样一个数据库，同时将 PHP 与数据库配置好

```
'mysql' => array(
 'driver' => 'mysql',
 'host' => 'localhost',
 'database' => 'iot',
 'username' => 'root',
 'password' => ' ',
```

```
'charset' => 'utf8',
'collation' => 'utf8_unicode_ci',
'prefix' => '',
)
```

当我们在前台创建一个数据后，可以在 MySQL 的后台查看：

| id | temperature | sensors1 | sensors2 | led1 | created_at          | updated_at          |
|----|-------------|----------|----------|------|---------------------|---------------------|
| 1  | 19.80       | 22.20    | 7.50     | 0    | 2013-12-31 07:03:59 | 2013-12-31 07:03:59 |
| 2  | 18.80       | 22.00    | 7.60     | 0    | 2013-12-31 07:03:59 | 2013-12-31 07:03:59 |