西安邮电大学

毕业设计（论文）

题目：基于云服务的物联网平台设计

学院：自动化

专业：自动化

班级：1403 班

学生姓名：郭子瑞

学号：06141079

导师姓名：李育贤 职称：教授

起止时间：2017 年 12 月 5 日 至 2018 年 6 月 10 日

毕业设计（论文）声明书

本人所提交的毕业论文《基于云服务的物联网平台设计》是本人在指导教师指导下独立研究、写作的成果，论文中所引用他人的文献、数据、图件、资料均已明确标注；对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式注明并表示感谢。

本人完全理解《西安邮电大学本科毕业设计（论文）管理办法》的各项规定并自愿遵守。

本人深知本声明书的法律责任，违规后果由本人承担。

论文作者签名：

日期：　　年　月　日

西安邮电大学本科毕业设计(论文)选题审批表

<table>
<tr><td>申 报 人</td><td>李育贤</td><td>职　　称</td><td colspan="2">教授</td><td>学　　院</td><td colspan="3">自动化</td></tr>
<tr><td>题目名称</td><td colspan="8">基于云服务的物联网平台设计</td></tr>
<tr><td>题目来源</td><td>科研</td><td colspan="3"></td><td>教学</td><td></td><td>其它</td><td>√</td></tr>
<tr><td>题目类型</td><td>硬件
设计</td><td>√</td><td>软件
设计</td><td></td><td>论文</td><td></td><td>艺术
作品</td><td></td></tr>
<tr><td>题目性质</td><td colspan="2">应用研究</td><td colspan="2">√</td><td colspan="2">理论研究</td><td colspan="2"></td></tr>
<tr><td>题目简述</td><td colspan="8">物联网是指通过各种信息传感设备，实时采集任何物体或过程的各种信息，与互联网结合形成的一个巨大网络，实现物与物、物与人，所有的物品与网络的连接。</td></tr>
<tr><td>对学生知识与能力要求</td><td colspan="8">掌握网络编程及应用，理解云服务基础架构原理；掌握微机原理与接口技术及单片机原理及应用,具有一定的模拟电路技术基础及网络通信原理,熟悉常用传感器原理及使用方法,具有嵌入式解决实际问题的动手能力。</td></tr>
<tr><td>具体任务以及预期目标</td><td colspan="8">实现基于云服务的物联网平台服务，实现物接入功能，即设备可进行连接建立及验证；物解析功能，实现基于 Protobuf 的数据解析服务；实现基础物联备设 SDK，设备可通过网络发送实时传感器数据。</td></tr>
<tr><td>时间进度</td><td colspan="8">2017 年 12 月 05 日—2017 年 12 月 10 日 选取毕设题目
2017 年 12 月 11 日—2018 年 01 月 06 日 查阅资料,撰写提交开题报告
2018 年 01 月 07 日—2018 年 03 月 04 日 确定系统架构设计方案 2018 年 03 月 05 日—2018 年 03 月 31 日 系统软件程序设计
2018 年 04 月 01 日—2018 年 04 月 15 日 软硬件的联合调试
2018 年 04 月 16 日—2018 年 04 月 31 日 系统功能完善
2018 年 05 月 01 日—2018 年 05 月 25 日 撰写毕业设计论文
2018 年 05 月 26 日—2018 年 06 月 01 日 修改、装订论文
2018 年 06 月 02 日—2018 年 06 月 10 日 准备毕业答辩</td></tr>
<tr><td>系（教研室）主任签字</td><td colspan="3">年　月　日</td><td>主管院长签字</td><td colspan="4">年　月　日</td></tr>
</table>

西安邮电大学本科毕业设计（论文）开题报告

学生姓名	郭子瑞	学号	06141079	专业班级	自动 1403
指导教师	李育贤	题目	基于云服务的物联网平台设计		

选题目的（为什么选该课题）

从我国 2009 年提出物联网发展战略以来，物联网在工业监控、城市管理、智能家居、智能交通等多个领域发展迅速。云计算是物联网发展的基石，也是实现物联网的核心，使物联网中以兆计算的各类物品的实时动态管理和智能分析变得可能，也促进了物联网和互联网的智能融合。

前期基础（已学课程、掌握的工具，资料积累、软硬件条件等）

大学已学如下课程：电子电路、传感器技术概论、TCP-IP 协议、嵌入式系统、标准与中间件技术等； 掌握 Eclipse，Keil，Python 等软硬件集成开发工具； 软硬件条件：具备笔记本一台供开发之用，拥有两台云端服务器和其他云服务资源做为业务服务点。

要研究和解决的问题（做什么）

物联网云服务平台由云服务器、终端组成，终端由控制器、以太网模块、输出接口和传感器接口组成。 物联网服务云平台整体工作流程如下: 1.本地传感器获取到所需的外部环境信息并传输到本地服务器； 2.本地服务器（个人笔记本）对传感器传输的信息进行处理，以 tcp 或 HTTP 的方式发送到云端服务器； 3.云端服务器对接收到的数据进行储存和展示； 4.用户端通过网络访问云端服务器的 WEB 界面，获取到感应器获取到的实时信息，并可查询过往历史信息。

工作思路和方案（怎么做）

利用温度传感器接收外界温度信息，并转换成机器可识别的机器信息,传输到担当上位机的本地服务器。个人笔记本暂代本地服务器，对传感器的数据进行转化并发送到服务器。两台 1 核 CPU1G 内存的 linux 云服务器，通过云服务提供的负载均衡器完成对服务器的抗压和避免单点故障。WEB 服务器配置 Nginx 完成对用户请求的代理转发，配置包括但不限于对爬虫，流量等多种现今常见的网络攻击的防护，配置对单个 IP 访问单位时间访问次数限制保证 Nginx 业务正常。Nginx 有很强的代理功能，但是一台 Nginx 就形成了单点，现在使用 keepalived 来解决这个问题，keepalived 可以实现故障转移切换，实现后端的健康检查，前端的高可用，确保了网站业务的正常运行。

指导教师意见

签字： 年 月 日

西安邮电大学毕业设计 (论文)成绩评定表

<table>
<tr><td>学生姓名</td><td>郭子瑞</td><td>性别</td><td>男</td><td>学号</td><td>06141079</td><td>专业班级</td><td>自动 1403</td></tr>
<tr><td>课题名称</td><td colspan="7">基于云服务的物联网平台设计</td></tr>
<tr><td>指导教师意见</td><td colspan="7">（从开题论证、论文内容、撰写规范性、学习态度、创新等方面进行考核）

评分（百分制）：　　指导教师(签字)：_____　　年　月　日</td></tr>
<tr><td>评阅教师意见</td><td colspan="7">（从选题、开题论证、论文内容、撰写规范性、创新和预期成果等方面进行考核）

评分（百分制）：　　评阅教师(签字)：_____　　年　月　日</td></tr>
<tr><td>验收小组意见</td><td colspan="7">（从毕业设计质量、准备、操作情况等方面进行考核）

评分（百分制）：　　验收教师(签字)：_____　　年　月　日</td></tr>
<tr><td>答辩小组意见</td><td colspan="7">（从准备、陈述、回答、仪表等方面进行考核）

评分（百分制）：　　答辩小组组长(签字)：_____　　年　月　日</td></tr>
<tr><td>评分比例</td><td colspan="7">指导教师评分　(20%)评阅教师评分　(30%)验收小组评分　(30%)答辩小组评分　(20%)</td></tr>
<tr><td>学生总评成绩</td><td colspan="2">百分制成绩</td><td></td><td colspan="2">等级制成绩</td><td colspan="2"></td></tr>
<tr><td>答辩委员会意见</td><td colspan="7">毕业论文(设计)最终成绩(等级)：

学院答辩委员会主任(签字、学院盖章)：　　年　月　日</td></tr>
</table>

目　录

摘　要

随着物联网技术和云服务技术不断发展，基于云服务的物联网技术应用也在不断增加。云计算作为物联网发展的基石，也是实现物联网的核心，使物联网中以兆计算的各类物品的实时动态管理和智能分析变得可能，也促进了物联网和互联网的智能融合。但传统网络的问题在物联网环境下仍有可能出现。所以合理利用云服务提供的资源，建立一个能够承受一定程度网络流量和常规网络攻击的适用于中小型企业的专业物联网平台很有必要。

本文以 DHT11 温湿度传感器来模拟物联网数据信息采集部分，以本地虚拟机模拟云端服务器。提出较为具体的设计方案并搭建相应架构。确保该架构具有一定的抵抗常规网络攻击和大流量的能力，确保该架构具有高可用性且易于维护和升级。

其次软件部分以 Python 为主语言，完成对传感器定时收集到的信息的处理并上传储存到云端数据库，并在云服务段完成对数据的显示。最终实现在互联网下，用户可以通过手机或个人电脑等设备以访问 WEB 的形式实时获取到传感器所采集到的信息。

关键字：云服务；物联网；高可用性能架构

ABSTRACT

With the continuous development of Internet of Things technologies and cloud service technologies, the application of Internet of Things technologies based on cloud services is also increasing. As the cornerstone of the development of the Internet of Things, cloud computing is also the core of the realization of the Internet of Things. This enables real-time dynamic management and intelligent analysis of various items in the Internet of Things in terabytes of computing and promotes the intelligent integration of the Internet of Things and the Internet. However, the problems of traditional networks may still appear in the Internet of Things environment. Therefore, it is necessary to properly use the resources provided by cloud services to build a professional Internet of Things platform suitable for small and medium-sized enterprises, able to withstand a certain degree of network traffic and regular network attacks.

This article uses the DHT11 temperature and humidity sensor to simulate the data acquisition part of the Internet of Things, and the local virtual machine simulates the cloud server. Put forward a more specific design plan and build a corresponding framework. Ensure that the architecture has the ability to resist regular network attacks and large traffic, ensuring that the architecture is highly available and easy to maintain and upgrade

Secondly, the software part uses Python as the main language to complete the processing of the information collected by the sensor at regular time, upload it to the cloud database, and complete the display of data in the cloud service segment. In the end, under the Internet, users can access information collected by sensors in real time through WEB access via mobile phones or personal computers.

Keywords: Cloud Services;Internet of Things;High-Availability Performance Architecture

第一章 引言

1.1 研究背景

自 2009 以来物联网发展战略提出以来，物联网在工业监控、城市管理、智能家居、智能交通等诸多领域得到了迅速发展。作为物联网发展的基石，云服务也是物联网的核心。它使得对物联网中各种物品的实时动态管理和智能分析成为可能。它也促进了物联网和互联网的智能集成。

1.2 发展趋势

云服务使传统 IT 产业结构在技术、商业模式和服务等方面发生了重大变化。服务提供商利用云服务为用户提供信息技术服务，使用户能够快速有效地通过网络获取自己所需的资源和服务，这对传统制造业产生了巨大的影响。所有的制造商都升级和改造他们的产品。同时，所有的产品都可以组成一个系统，可以为用户提供一个完整的服务，而不是单一的产品，并大力建立自己的移动互联网生态系统。对于许多硬件搭建的企业来说，互联网的部署和运营是一场噩梦，缺乏专业的研发团队，平台运行维护的成本和精力，以及保障安全问题的难度都是难以解决的。因此，建立适合中小企业的专业云服务平台架构，可以解决这些硬件厂商的困难，不仅使制造商对原有产品进行简单、低成本的改造，而且使 USE 的定制服务由云服务平台提供，以增强用户体验，而不必担心未来的产品。提升和升级结构也将有助于迅速占领市场份额。

物联网云平台架构，主要由传感器部分，数据上传部分，云端服务器部分和 WEB 端的用户访问部分组成。将物联网和云服务合理结合，既减少中小企业在网络上面的维护成本，也可以提高用户的体验。

本次毕业设计核心是围绕着云服务器，合理利用其提供的资源搭建一套能够抵御一定流量和部分网络攻击的架构，且该架构需要具有高可用性，易于维护和升级的特性，并使该机构可以接受并处理来自传感器的有效信息。

第二章 物联网云平台的整体架构设计

2.1 架构整体需求分析

本次物联网云平台设计是基于本地 Linux 虚拟机模拟 Linux 云服务器来完成的。其目的是为了完成一条相对成熟的适用于中小企业的物联网云端架构。对于该架构的需求点如下：

1）节约成本

对于中小企业来说不论是购买成熟的私有云部署方案还是自己搭建私有云架构体系都需要投入大量的资金和人力，所以选择现有的公有云并将自身的业务依托在其上面是相对合理的选择。但目前市面上的公有云服务基本是以配置来决定单位是的价格，所以合理使用资源节约成本势在必行。

2）架构简单稳定

对于中小企业来说，其雇佣专门的人员去管理云上的资源是不太可能，大多数情况下是有开发人员兼职运维管理，所以架构简单稳定且易于管理和升级必不可少。

3）数据信息的安全和储存

对于现今物联网云平台来说数据的价值远大于其他事物，而且现今大数据时代的到来使得有用的数据不在局限于实时数据，过去的历史数据也发挥这巨大的价值所以对数据的安全的保护和储存管理不能缺少。

4）可防御网络攻击

随着互联网的发展网络攻击也在不断怎多，虽然公有云服务可以帮助阻挡大部分攻击，使现今的网络攻击方式多集中在爬虫或者流量攻击，但仍然需要对其进行防护，所以该架构必须具有一定的防御网络攻击能力。

5）实时的数据获取和传输

对于物联网云平台来说数据的实时获取是基本功能，但要求获取到的数据稳定的传输到云端是需要考虑的问题。

2.2 架构整体设计方案

2.2.1 整体设计

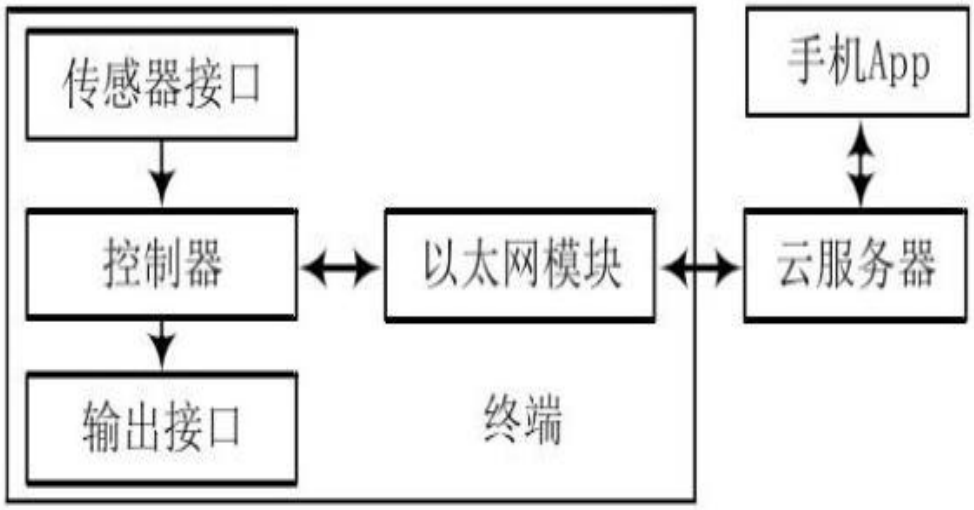

图 2.1　整体框架设计图

2.2.2 硬件设计

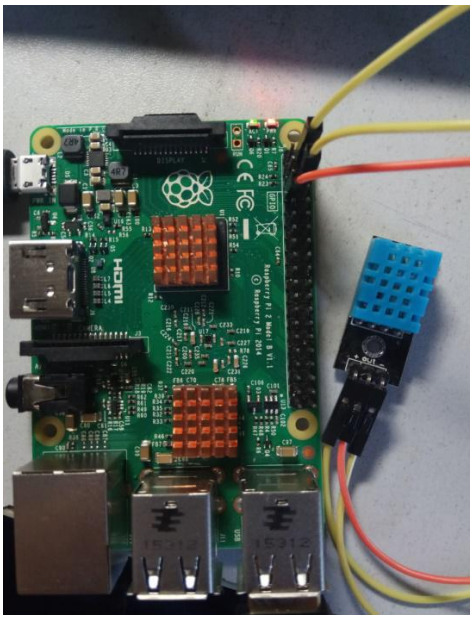

图 2.2　硬件设计图

2.2.3 云平台架构设计

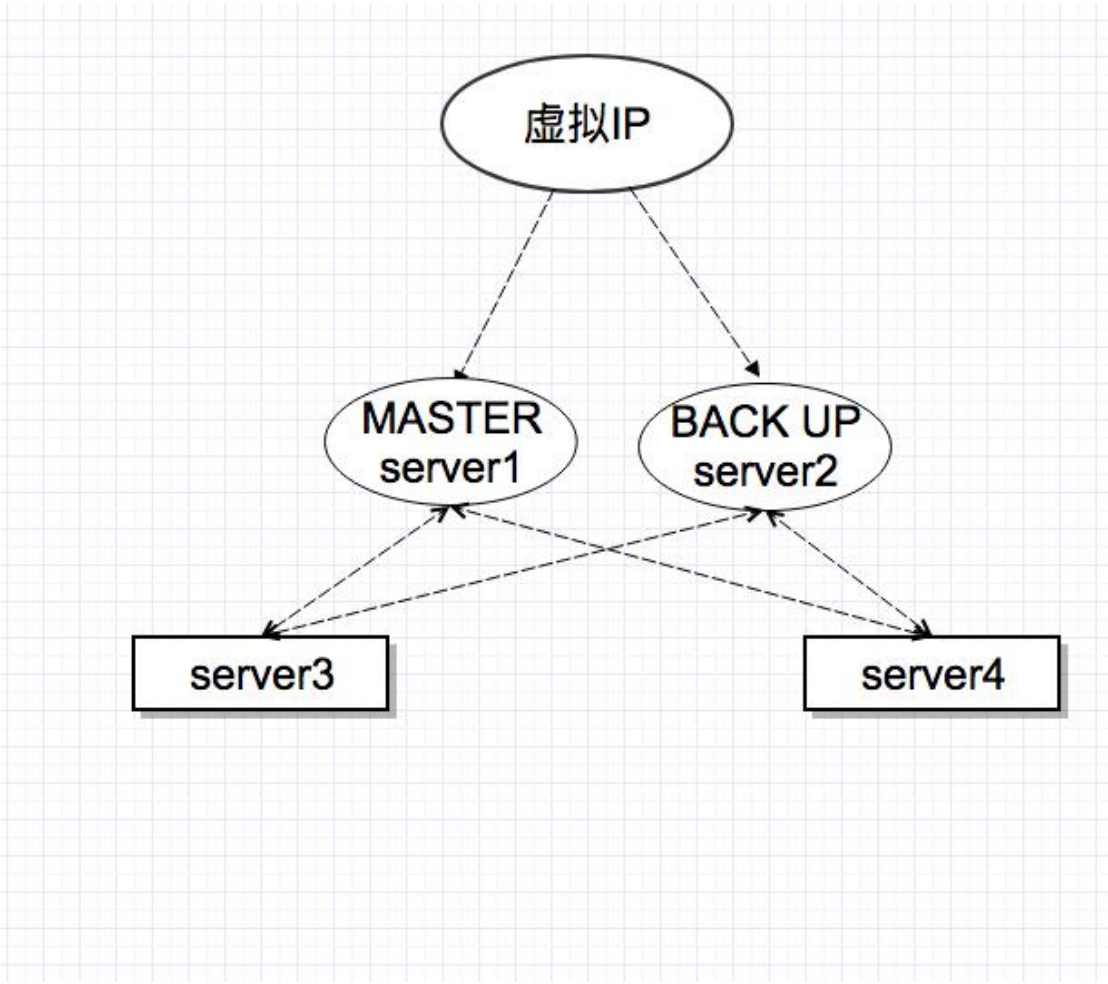

图 2.3　云端框架设计图

第三章 物联网云平台所用硬件选型

物联网云平台架构，主要由传感器部分，数据上传部分，云端服务器部分和WEB 端的用户访问部分组成。其中传感器部分和数据上传部分由硬件组成。由于该部分是模拟生产环境，所以选型方面可能与生产环境所需硬件设备有所差异，但核心理论体系和技术关键并不受影响。

3.1 传感器部分选型

传感器作为一种用于侦测环境中所生事件或变化，并将此消息发送出至其他电子设备（如中央处理器）的设备，其是物联网云平台的必要组件。在选型方面由于各企业在传感器方面都有不同的需求和用途，所以本文以温湿度传感器DHT11 作为器材，来模拟传感器的输出采集。

DHT11 型数字温湿度传感器是一种具有自校准数字信号输出的温湿度重合传感器。其应用专有的数字模块采集技术和温湿度传感技术都确保了其具有超快响应，抗干扰能力强，性价比较高等优点。而且 DHT11 温湿度传感器的体积小，功耗低，测量误差相对较小，故选择 DHT11 温湿度传感器为本次毕业设计的传感器部分的器材。

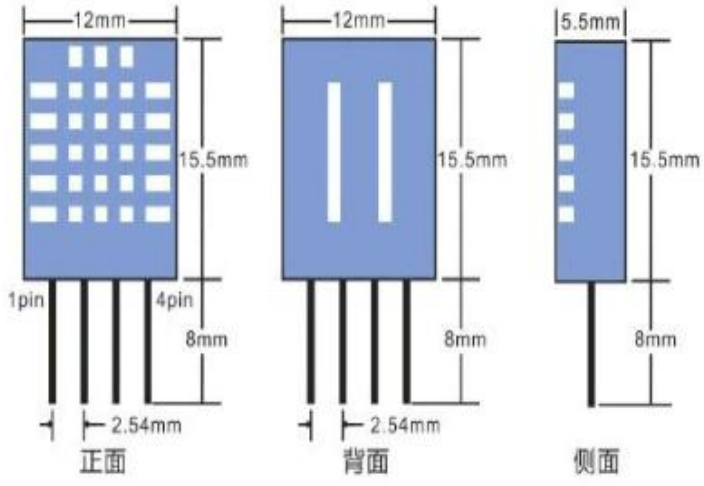

图 3.1　DHT11 器件图

表 3.1　DHT11 引脚说明图

Pin	名称	注释
1	VDD	供电 3-5.5VDC
2	DATA	串行数据，单总线
3	NC	空脚，请悬空
4	GND	接地，电源负极

3.2 数据上传部分选型

数据上传部分一般由传感器接口；控制器；输出接口和以太网模块构成。在这部分的选型方面有许多种优秀的选择方案，但由于该部分并非本次毕业设计的核心和重点所以在选型方面选择了集成度和便捷度相对较好的树莓派 2b+。

树莓派 2b+作为一款基于 Linux 的单片机电脑。其拥有一个四核心 900MHZ 的 CPU 和 1G 的内存这是的它可以胜任大部分情况下的数据处理工作，配置有 4 个 USB 接口使其拥有一定的扩展能力，配置有 SD 卡槽可以自由扩展系统的功能来完成相对复杂的工作，而且支持图形界面和命令模式这是的在代码调试是较为方便。在介入 wifi 模块后树莓派便可以与互联网通信，通过代码完成数据的上传工作。

第四章 物联网云平台软件设计

基于云服务的物联网平台设计的软件设计主要分为：1）数据收集处理部分；2）数据远程上传数据库部分；3）云端数据库数据调用部分；4）数据库数据处理及显示部分。这四个部分中，第一部分和第二部分的代码配置在本地终端上即由树莓派的系统来完成这两部分任务。第三部分和第四部分都在云端的服务器上即虚拟机来完成这两部分任务。

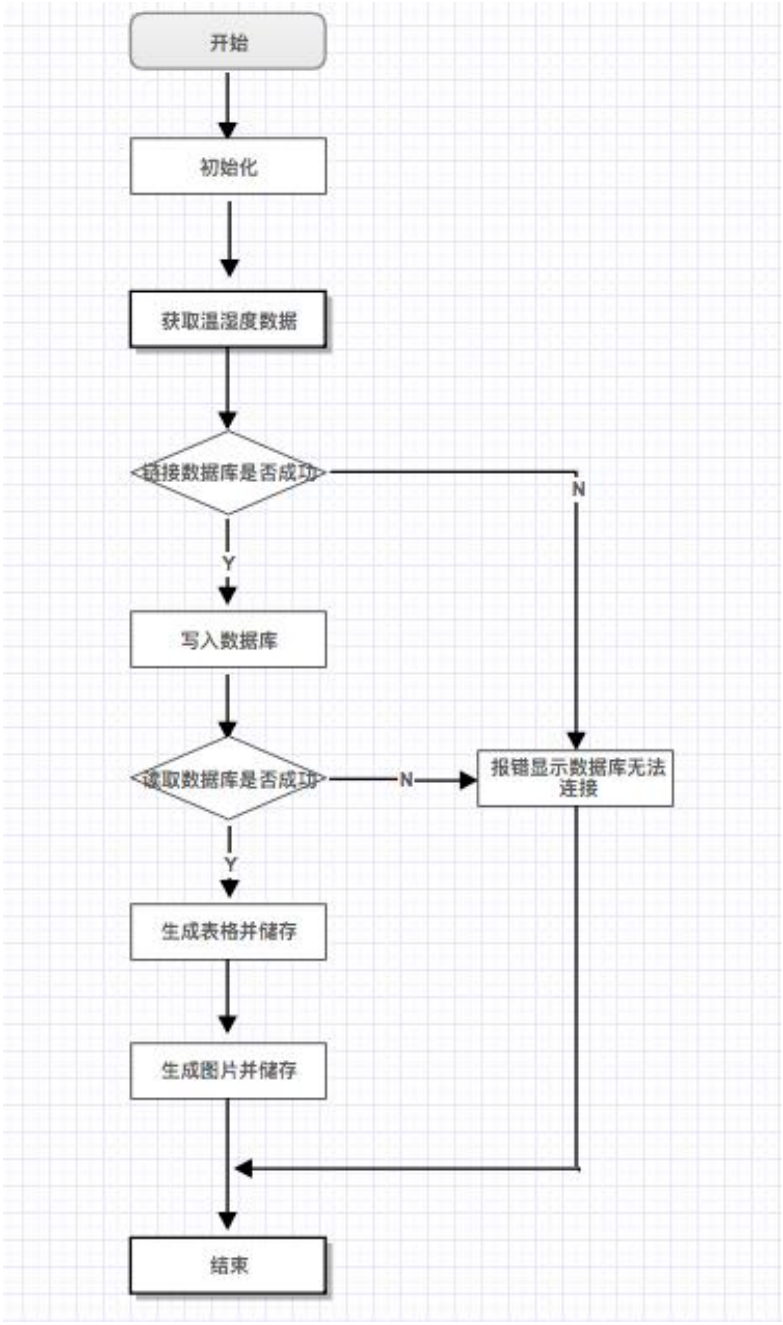

图 4.1　软件流程图

4.1 数据收集处理设计

利用 Python 代码使用第三方库 GPIO 库使得系统可以实时获取到 DHT11 温湿度传感器所获取到的温湿度信息，但由于 DHT11 使用的单线协议（1-wire）没有同步脉冲，所以需要对时序进行一定的设置。然后将处理后的信息准确有效的通过 Python 相应模块 MySQLdb 模块完成数据上传。

```
# coding=utf-8
#!/usr/bin/python
import RPi.GPIO as GPIO
import time
import MySQLdb


channel = 33#使用的GPIO口
data = []
j = 0

GPIO.setmode(GPIO.BOARD)#设定GPIO口编码格式

time.sleep(1)#延时1s

GPIO.setup(channel, GPIO.OUT)#设定GPIO口为输出
GPIO.output(channel, GPIO.LOW)#给GPIO口低电平
time.sleep(0.02)
GPIO.output(channel, GPIO.HIGH)#打开GPIO口
GPIO.setup(channel, GPIO.IN)#设定GPIO口为输入口

while GPIO.input(channel) == GPIO.LOW:
  continue
while GPIO.input(channel) == GPIO.HIGH:
  continue
#打开GPIO口，初始GPIO口为高电平


while j < 40:
  k = 0
  while GPIO.input(channel) == GPIO.LOW:
    continue
  while GPIO.input(channel) == GPIO.HIGH:
    k += 1#当GPIO口进来一个高电平时 k加一
    if k > 100:#k值到100结束
      break
  if k < 8:
    data.append(0)#k小于8时，在数组data放0
  else:
    data.append(1)#k大于8时，数组data放1
  j += 1
```

```
humidity_bit = data[0:8]
humidity_point_bit = data[8:16]
temperature_bit = data[16:24]
temperature_point_bit = data[24:32]
check_bit = data[32:40]

humidity = 0 #湿度
humidity_point = 0
temperature = 0#温度
temperature_point = 0
check = 0

for i in range(8):#二进制转化为十进制
  humidity += humidity_bit[i] * 2 ** (7-i)
  humidity_point += humidity_point_bit[i] * 2 ** (7-i)
  temperature += temperature_bit[i] * 2 ** (7-i)
  temperature_point += temperature_point_bit[i] * 2 ** (7-i)
  check += check_bit[i] * 2 ** (7-i)

tmp = humidity + humidity_point + temperature + temperature_point

if check == tmp:
  print "temperature :", temperature, "*C, humidity :", humidity, "%"
else:
  print "wrong"
  print "temperature :", temperature, "*C, humidity :", humidity, "% check :", check, ",
tmp :", tmp

GPIO.cleanup()
```

图 4.2　数据收集及处理设计

4.2 数据远程上传数据库设计

利用 Python 代码和云端数据库提供出的用户，密码以及数据库的地址。将树莓派系统与云端数据库之间的通信打通。并通过系统的定时任务将梳理过后的数据连同时间定时上传到云端的数据库。采用系统定时任务而非循环的原因在与，由于我们所需求的温湿度精确性和时间准确度不高，且采用循环的方式会导致 Python 程序一致处于运行状态会占用大量的 CPU 和内存，可能导致系统的崩盘。而采用系统定时任务的方式可以有效避免对系统内存和 CPU 的浪费节约资源成本。

```
conn = MySQLdb.connect(host='localhost',
                       port=3306,
                       user='root',
                       passwd='QAZQAZ',
                       db='test')
cur = conn.cursor()
value = [temperature,humidity,str(time.strftime('%A %X %Z',time.localtime(time.time())))]
cur.execute("insert into dht (dht_tem,dht_hum,dht_time) values(%s,%s,%s)",value)
conn.commit()
conn.rollback()
```

图 4.3　数据远程上传处理设计

4.3 云端数据库数据调用设计

同样采用系统的定时任务来完成对数据库数据的调用。这其中通过 Python 的 MySQLdb 模块和云端数据库提供的只读用户，密码以及数据库地址。查找并获取数据库内的部分数据，之所以不对整个数据库进行调用的原因在于：1）实际生产需求不会每次都调用所有数据，出特殊情况外，大部分时间只会调用就近时间的部分数据；2）调用整个数据库会给云服务的 CPU 和内存带来巨大的压力，浪费资源；3）调用整个数据库还可能会产生慢查询或链接池堆积导致业务不可用。

```
# coding:utf-8
import pymysql

conn = pymysql.connect(host='127.0.0.1', user='root', db='test1', charset='utf8')

cur = conn.cursor()

sql1 = """select 时间 from 温湿表 order by 时间 DESC """
sql2 = """select 温度 from 温湿表 order by 时间 DESC """
sql3 = """select 湿度 from 温湿表 order by 时间 DESC """
try:
    # 执行sql语句
    cur.execute(sql1)
    # 提交到数据库执行
    conn.commit()
except:
    # 如果发生错误则回滚
    conn.rollback()
time = cur.fetchall()  # 取所需数据

try:
    # 执行sql语句
    cur.execute(sql2)
    # 提交到数据库执行
    conn.commit()
except:
    # 如果发生错误则回滚
    conn.rollback()
temperature = cur.fetchall()  # 取所需数据
# print(temperature)

try:
    # 执行sql语句
    cur.execute(sql3)
    # 提交到数据库执行
    conn.commit()
except:
    # 如果发生错误则回滚
    conn.rollback()
humidity = cur.fetchall()  # 取所需数据

conn.close()

from prettytable import PrettyTable

```

图 4.4　云端数据库数据调用设计

4.4 数据库数据处理及显示设计

利用 Python 代码的 PrettyTable 模块对从数据库提取进行处理平输出成表格写入指定的 txt 文件内；利用 matplotlib 将表格信息制分别作成温度和湿度两张图片并储存在制定文件夹内，方便 HTML5 制作的 WEB 界面进行读取。且储存的图片和表格会同定时任务一起定时刷新，确保每次用户访问得到的信息都是最新过去到的数据。

```
from prettytable import PrettyTable

data = PrettyTable(["time", "temperature", "humidity"])
data.align["time"] = "r"  # 以name字段左对齐
data.align["temperature"] = "r"
data.align["humidity"] = "r"
data.padding_width = 1  # 填充宽度
# data.add_column()

list_time = []
list_temperature = []
list_humidity = []
for num in range(0, 10):  # 迭代 0 到  之间的数字
    if num <= 10:  # 确定第一个因子
        data.add_row([*time[num], *temperature[num], *humidity[num]])
        list_time = list_time + [*time[num]]
        list_temperature.extend([*temperature[num]])
        list_humidity.extend([*humidity[num]])
        num = num + 1  # 计算第二个因子
    else:  # 循环的 else 部分
        break
print(data)
with open('/usr/local/etc/nginx/test.txt', 'wt') as f:
    print(data, file=f)

import matplotlib.pyplot as time_temperature  # 导入模块

time_temperature.plot(list_time, list_temperature, linewidth=1)  # 传入列表 linewidth决定绘制线条的粗细
time_temperature.title('temperature', fontsize=24)  # 标题
time_temperature.xlabel('time', fontsize=14)
time_temperature.ylabel('temperature', fontsize=14)
time_temperature.tick_params(axis='both', labelsize=14)
for a, b in zip(list_time, list_temperature):
    time_temperature.text(a, b + 0.1, '%.0f' % b, ha='center', va='bottom', fontsize=7)
# time_temperature.show()  # 输出图像
time_temperature.savefig('/tmp/temperature.png')
time_temperature.show('/mnt')  # 输出图像

import matplotlib.pyplot as time_humidity  # 导入模块

time_humidity.plot(list_time, list_humidity, linewidth=1)
time_humidity.title('humidity', fontsize=24)
time_humidity.xlabel('time', fontsize=14)
time_humidity.ylabel('humidity', fontsize=14)
time_humidity.tick_params(axis='both', labelsize=14)
for a, b in zip(list_time, list_humidity):
    time_humidity.text(a, b + 0.1, '%.0f' % b, ha='center', va='bottom', fontsize=7)
# time_humidity.show()
time_humidity.savefig('/tmp/humidity.png')
```

图 4.5　数据库数据处理及显示设计

第五章 物联网云平台架构选型

作为一套可移至且方便维护的物联网云平台架构，其架构设计是整套体系的核心和关键。为保证整套系统具有抵御大流量和部分攻击的能，且要易于维护和升级。这套系统的架构一定要相对简单且具有高可用性。所以整个物联网云平台的选型包括一下几点：1）系统选型；2）数据库选型；3）WEB 层服务选型；4）高可用方案选型 5）架构选型。

5.1 系统选型

目前共有云平台提供的服务器系统主要有：Windows Server；Ubuntu; CentOS;Red Hat Enterprise Linux 等。其中除去 Windows Server 外都是 Linux 系统。在系统选型方面需要考虑的网面包括但不限于：1）上手难易程度；2）运维成本和部署成本；3）市场占有率等多方因素。

Ubuntu 是一个流行的 Linux 版本，它基于强化 Debian 的 unstable 版本而来的，并以“最好的 Linux 桌面系统”出名。近年来 Ubuntu 也推出了 Ubuntu Enterprise Linux，其在企业级 Linux 应用市场的份额也得到了很大的提高。虽然其专业的技术支持和技术社区相较于 RHEL 较弱，但其优秀的桌面系统会极大的降低开发人员的入门门槛，而在安全防护方面的更新与 RHEL 并没有太大差异。所以 Ubuntu 系统对于中小企业来说是一个较为优秀的选择。所以在本次毕业设计中选择的系统就是 Ubuntu 系统。

5.2 数据库选型

由于本次毕业设计是基于共有云服务来完成的，所以在数据库的选型方面只会考虑云服务商能够提供的几种主流数据库来进行选型。其中包括：1）Oracle；2）MySQL；3）SQL Server；4）PostgreSQL。而在数据库选型方面需要考虑的问题包括：1）安全性；2）操作；3）使用风险；4）性能；5）易维护性和价格。

MySQL 数据库是一个关系型数据库管理系统。它现今属于甲骨文公司。

MySQL 在 WEB 应用方面 MySQL 是最好的 RDBMS 应用软件之一。MySQL 是一种关联的数据库管理系统，它将数据存储在不同的表中，而不是把所有的数据放在一个大的仓库中，这样可以提高速度和提高灵活性。MySQL 所使用的 SQL 语言是用于访问数据库的最常用的标准化语言。而且该类型数据库体积小、速度快、成本低、开源，是开放人员最常用的数据库之一。所以相对于 MySQL 存在的不支持热备份，无法自定义数据类型等缺点。MySQL 数据库在本次毕业设计中的优势十分明显，故选择 MySQL 数据库作为本次架构设计的数据库。

5.3 WEB 层选型

由于本次使用的系统为 Linux 系统，所以主流的 WEB 层软件只有两种，Nginx 和 Apache。两个的有确定也十分明确，Apache 更适合去处理动态请求且模块多样化可以满足需求，运行稳定且 bug 较少。但同样的 Apache 相较于 Nginx 更加的笨重且在处理静态请求方面不尽如人意，而且配置较为复杂。而 Nginx 作为一款轻量级的 WEB 服务其拥有极好的处理高并发和静态请求的能力，而且占用的服务器资源较少并且配置相对简单。但同样的在处理动态请求方面 Nginx 的表现十分差劲而且功能不算完善。结合本次毕业设计，物联网云平台对外的 WEB 界面是一个纯静态界面不存在动态请求，而且由于不存在专门的运维人员所以相对容易配置且拥有庞大技术社区的 Nginx 更具有优势，所以 WEB 层服务的选型选择 Nginx。

5.4 高可用性方案选型

高可用性的核心十分简单那便是冗余，利用其他服务器作为关键节点的备用设备，使其在崩溃后服务可以继续域名。本次毕业设计的核心便是设计一套易于部署和升级的基于云服务的物联网平台架构，由于该架构需要具有的多种特性所以高可用性的方案选择十分重要。而现今较为流行方案用两种分别是 keepalived 和 heartbeat。

两者的优缺点也十分的明确，虽然通过软件配置，可以从一台失败的计算机向另一台正常运行的计算机提供资源（例如 IP 和程序服务等）。但在应用场景和特点上各有不同。其中 heartbeat 支持更多的功能，在大型的进群架构上 heartbeat 有种人不俗的表现，但其的配置相对复杂而且在 heartbeat2.1.4 后，更是将自身拆分成了多个小系统，这无疑增加了在故障时的运维成本。而 keepalived 从体量和配置维护的角度来说都相对适用于没有专职运维人员的企业但其支持的功能较少和通信的可靠性不如 heartbeat 是不争的事实。结合本次毕业设计，我们需要的是一种轻量级的方便配置的高可用方案，只要具有基本功能和相对稳定的通信模式就能够满足要求。所以在高可用方面的选型选择 keepalived。

5.4 架构选型

目前市面上常见的网站架构有单机模式，单点负载均衡模式，高可用性可扩展集群模式。单机模式所使用的资源最少，但一旦节点发生故障则整个服务瘫痪，这对于用户体验来说极差且无法承载大流量的访问，当然也可以通过提高服务器配置来提高瓶颈，但这样做的费用极高且无法根本上解决问题。单点负载均衡模式可以有效避免单机模式的单点故障而服务崩溃的情况，由于其负载均衡的特性当一个节点由于故障而崩溃时，调度服务器会自动跳过故障节点保证访问的正常进行。但单点负载均衡模式的调度服务器也存在瓶颈和故障的可能，一旦调度服务器崩溃其结果是不论访问节点是否正常，该服务都会崩溃。而高可用性可扩展集群模式则是单点负载均衡模式的进一步升级，在访问端采用和单点负载均衡模式一样负载均衡方式有效分流较少单台服务器压力，而在调度服务器上采用双机热备的方式保证当主调度节点崩溃是，备用节点快速接管任务保证服务正常进行，这种方式也是目前大型网站常用的架构方式，其在访问服务端有着可一快速平面扩展的优势切在调度端可以承载大流量的工作而且并不需要太高的配置。

在本次毕业设计中，我们利用两台虚拟机通过 keepalived 使它们实现双机热备作为流量的调度端，使其实现在调度端的冗余，保证在主调度端出现崩溃后，

流量调度端快速切换到备用节点，完成对与用户的无感知切换，保证服务的正常运行。现今解决突发事件导致的节点故障的方式大多数都是利用冗余的方式，虽然这种方式存在一定的资源的浪费，但其可以有效的保证在承受攻击时，可以为工作人员赢得更多的抢救时间，不至于在故障发生后直接导致服务不可用，这极大的挺高了用户体验。并且在调度端的 keepalived 服务器上开放邮件发送功能，即在节点发生故障导致节点更换时，及时发送邮件通知管理员这样可以保证在面对持续大流量的访问时可以有所准备。利用 keepalived+lvs 的架构体系中 lvs 部分实现对流量向后端的分流时后端的 Nginx 服务器。

后端由两台 Nginx 服务器来完成对请求的接受，由于此次毕业设计的 WEB 界面都是静态资源的所以对两台 Nginx 服务器的访问采用轮询的方式，保证两台服务器均摊流量压力，且一旦一台 Nginx 服务器崩溃后，另一台 Nginx 服务器还可以继续保证服务正常运行。在平台的横向扩容和应急清酒方面的可以使用快速的克隆两台 Nginx 服务器中正常运行的一台，将其加入负载均衡中来完成对服务器的应急抢救和在大流量来临前的提前冗余准备。两台 Nginx 服务器利用 Python 脚本直连远端数据库，利用定时任务将数据库内的数据生成图片并储存在本地作为静态资源。

第六章 物联网云平台架构部署

根据对物联网云平台架构个部分的选型结果，此次部署将使用 Ubuntu 16.04 的系统作为四台虚拟机的系统。其中一台虚拟机（MySQL）部署 MySQL 数据库，其他三台虚拟机（Server1,Server2,Server3）部署相关服务。此次部署 IP 分配如下表所示。

表 6.1　IP 分布及说明

IP	所在服务器	说明
192.168.0.100	Server1 或 Server2	虚拟 IP（VIP）
192.168.0.120	MySQL	数据库真实 IP
192.168.0.121	Server1	Server1 真实 IP
192.168.0.122	Server2	Server2 真实 IP
192.168.0.123	Server3	Server3 真实 IP
192.168.0.124	Server4	Server4 真实 IP

6.1 MySQL 数据库的部署

数据库作为此架构唯一的输出储存点，所以其安全性需要很高。所以在本次毕业设计中采用双用户的方式进行管理。即树莓派上使用的写入用户和虚拟机对数据库的读取用户不是同一用户且权限大小也不同。其中数据写入用户也只具有对表的写入，读取，更新等常用权限而不具备对库的操作权限和对表的删除权限。而读取数据库的用户更是只具有对表的读取权限。

在虚拟机 MySQL 上利用 apt-get 命令安装 MySQL 数据库，更改配置文件 /etc/mysql/mysql.conf.d/mysql.cnf 使数据库可以被其他 IP 访问。重新启动数据库，刷新配置是更爱的配置生效。从本地利用 root 用户登陆数据库并创建专用库 ‘temperature_humidity’，进入库并创建表‘dht’。创建用户‘t_h_rw’赋予用户 ‘temperature_humidity’库的‘Select’, ‘Insert’, ‘Update’， ‘References’，‘Create’权限

并只允许该用户从制定 IP 段访问数据库。创建用户‘t_h_r’赋予用户‘temperature_humidity’库的‘Select’权限并只允许该用户从制定 IP 段访问数据库。

6.2 Nginx 服务部署

本次选在使用 Nginx 这种轻量级的 WEB 服务，主要是以为定时任务的 Python 代码已经将结果以图片及表格的方式输出了，Nginx 只需要去读取静态结果。在虚拟机 Server3，Server4 上利用命令 apt-get 安装 Nginx 插件，修改配置文件。

限制单个 IP 的单位时间内的访问次数，如果超过该次数则将该 IP 加入黑名单 24 小时。在使用公有云服务的情况下，常见的攻击便是利用多个 IP 对网站进行 DDoS 攻击，所以显示 IP 的访问次数可以有效降低 DDoS 攻击对网络资源的占用，从而减少服务器因攻击而崩溃的几率。配置 rborts.txt 文件，使常规爬虫无法爬取该网站的相关资源。由于现今网上大多数爬虫都带有有几大互联网厂商的标签，所以禁止爬虫访问可以有效保护网站数据安全。可以在开启配置协议的强制跳转，但由于本次毕业设计并没有 HTTPs 所需的证书，古没有开放 443 端口，只是用了 HTTP 协议，由于域名需要 dns 的解析才能生效所以在本次毕业设计中不仅有对域名的监听也有对虚拟 IP 的监听。最后重新启动 Nginx，并查看端口是否开放。

6.3 keepalived 服务部署

此次高可用性服务部署选则了，目前开源情况下最常用的方案 keepalived+lvs 去实现架构的高可用和访问的轮询并实现双机热备。即在 master 节点由于故障后服务立刻有 back up 节点接管，保证服务的正常运行，其变更对于用户的访问来说是无感知的，当 master 节点的故障被排除并重新上线后，服务回归之 master 节点管理。而在用户访问时，用户每次访问的并不一定是同一台服务器，而是两台内容完全一致的服务器在被轮流访问。

在虚拟机 Server1，Server2 上利用 apt-get 命令那装 keepalived 插件。在

/etc/keepalived/keepalived.conf 文件内配置相关功能。1）配置在 keepalived 在发生切换时需要给指定目标发送邮件，一般情况下 keepalived 是不会切换的。只有在 master 节点崩溃或者恢复时才会发生切换，所以这个时间点发邮件可以有效提醒到开发人员。2）配置两个节点的相互认证，设置合理检查时间放置节点崩溃而另一个节点没有作出响应。3）配置虚拟 IP，使 IP192.168.0.100 处于可以被链接状态并且对外开放制定端口，且配置 lvs 的调度算法，在本次毕业设计中使用轮询，转发规则选择 DR，使用协议选择 TCP。4）配置后端真实的服务器 IP，Server3，Server4 都在其中，且明确状态和超时时间及发现超时后重连次数和重连的间隔时间。

更改完成后，启动 Server1，Server2 上的 keepalived 服务。并使用 ip addr 命令查看虚拟 IP 是否出现在 master 节点上且其 80 端口可以被访问。利用命令 ipvsadm –ln 查看轮询模块的配置 Server3，Server4 是否全在其中且权重和活跃程度相同。

第七章 物联网云平台架构测试

7.1 数据库模块测试

数据库模块作为此次毕业设计的承接模块起的安全性和可用性是测试的关键，数据库模块的正常运行及可以保证数据的价值不会消失也保证了整个平台的正常工作。

本模块测试是为了保证数据库上传部分和云端架构可以有效的链接入数据库，且除了制定 IP 外的其他 IP 无法通过数据库已有用户远程连接进入数据库。保证制定用户只具有应有的权限且可以正常使用相关权限进行工作。测试结果如下图。

7.2 高可用性及轮询模块测试

高可用性模块和轮询模块是用户直接访问的模块，也是这次毕业设计的核心模块。正常状况下轮询模块的工作对于用户是无感知的，但为了能够确定轮询模块是否正常工作，故需要对用户访问的界面进行一定的更改以确保轮询模块可被感知。其次是高可用性模块，该模块的测试比较简单，及不管是单纯的 keepalived 服务崩溃还是其所在的整个虚拟机崩溃都不会形影到用户的正常访问。

7.2.1 轮询模块的测试

单纯的轮询模块测试，只需要确保所有访问都被从虚拟 IP 依次转发到下属的服务器上。而当有虚拟机的 Nginx 节点发生故障时会自动跳股对故障姐点的访问，只访问正常节点。由结果可以轮询访问正常运行，流量经过 lvs 处理平均输出到两台 Nginx 服务器上。

```
root@ubuntu:/home/tom# curl 192.168.42.100
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>th.xupt.com</title>
</head>
<body>
        <h1>temperature_humidity_3</h1>

        <img src="humidity.png" width="1000" height="500" />
        <img src="temperature.png" width="1000" height="500" />
</body>
</html>
root@ubuntu:/home/tom# curl 192.168.42.100
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>th.xupt.com</title>
</head>
<body>
        <h1>temperature_humidity_2</h1>

        <img src="humidity.png" width="1000" height="500" />
        <img src="temperature.png" width="1000" height="500" />
</body>
```

图 7.1　轮询测试图

7.2.2 高可用性模块测试

高可用性模块测试分为两个部分分别是单纯的keepalived故障和虚拟IP工作节点的整体崩溃。正常情况下的处理结果应该是不论是哪种故障发生，虚拟 IP 都应当会快速转移到 back up 节点并正常工作。但当 master 节点恢复并再次上线后，虚拟 IP 会快速的从 back up 节点转移回到 master 节点，且访问的轮询正常进行。有结果来看当 master 节点崩溃后虚拟 IP 快速转移到了 back up 节点，且轮询访问仍然可以进行，与预测结果一致。

```
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
 default qlen 1000
    link/ether 00:0c:29:92:10:ac brd ff:ff:ff:ff:ff:ff
    inet 192.168.42.226/24 brd 192.168.42.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet 192.168.42.100/32 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe92:10ac/64 scope link
       valid_lft forever preferred_lft forever
root@ubuntu:/home/tom#
```

图 7.2　虚拟 IP 漂移图

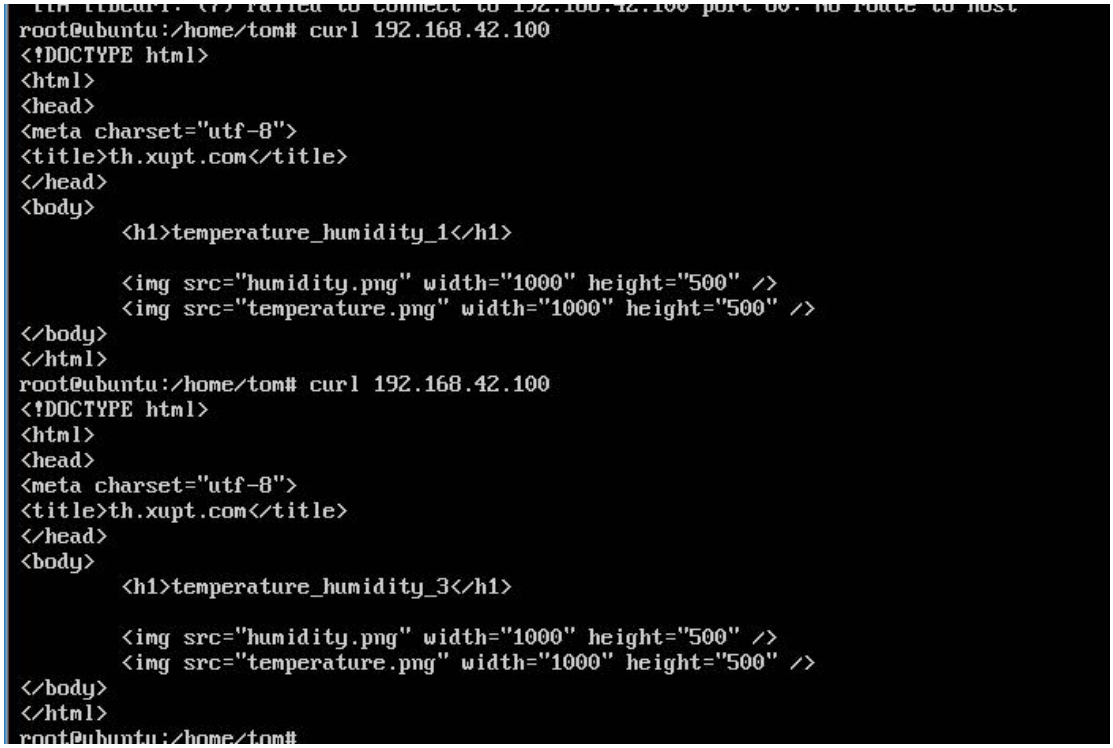

图 7.3　节点崩溃后浏览结果图

7.3 应用测试

作为一个面向用户的系统平台架构，该系统应当能承受一定的流量访问，故需要对整体框架进行压力测试，保证响应时间和服务器的状态。而且本次毕业设计的 WEB 界面也应当可以支持多种浏览器的访问和手机的访问。所以本次应用测试分为压力测试和页面访问测试。

7.3.1 压力测试

压力测试方面选择使用 Ab（Apache Bench）命令进行压力测试。Ab 命令作为一款小巧且可以精确控制请求次数的压测方式，其可以显示很多详细的信息，本次压测分别选择 1000RPS 和 3000RPS 的请求量，要求不能出现失败请求且不论是调度节点还是 Nginx 服务器的 cpu 和内存部门出现过高波动。

```
Concurrency Level:      100
Time taken for tests:   0.337 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      514000 bytes
HTML transferred:       257000 bytes
Requests per second:    2964.68 [#/sec] (mean)
Time per request:       33.730 [ms] (mean)
Time per request:       0.337 [ms] (mean, across all concurrent requests)
Transfer rate:          1488.13 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        6   17  17.8     12      80
Processing:     4   15  11.7     13      88
Waiting:        1   10  10.5      7      81
Total:         12   33  20.4     24      96
```

图 7.4　1000RPS 压测结果图

```
Concurrency Level:      500
Time taken for tests:   0.798 seconds
Complete requests:      3000
Failed requests:        0
Total transferred:      1542000 bytes
HTML transferred:       771000 bytes
Requests per second:    3758.73 [#/sec] (mean)
Time per request:       133.024 [ms] (mean)
Time per request:       0.266 [ms] (mean, across all concurrent requests)
Transfer rate:          1886.71 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       34   56  14.4     53      91
Processing:     4   65  16.2     66     118
Waiting:        2   24  17.1     21      93
Total:         87  121   6.6    122     155
```

图 7.5　3000RPS 压测结果图

从测试结果图来看不论是 1000RPS 还是 3000RPS 都没有出现失败请求，但由于这次压测试做的内网压测所以其响应时间不具备参考性。而观察调度服务器和 Nginx 服务器的 CPU 都有所提升但处于可控范围内的，故压测结果合格。

7.3.2 页面访问测试

由于需要对页面显示进行查看，所以本次测试中选择了几款典型的浏览器包括：谷歌浏览器，IE 浏览器，在手机端也是直接访问该界面，看是否会出现图片不适配的情况。需要保证浏览器访问不会报错，且图片清晰表格是否可以正常显示。

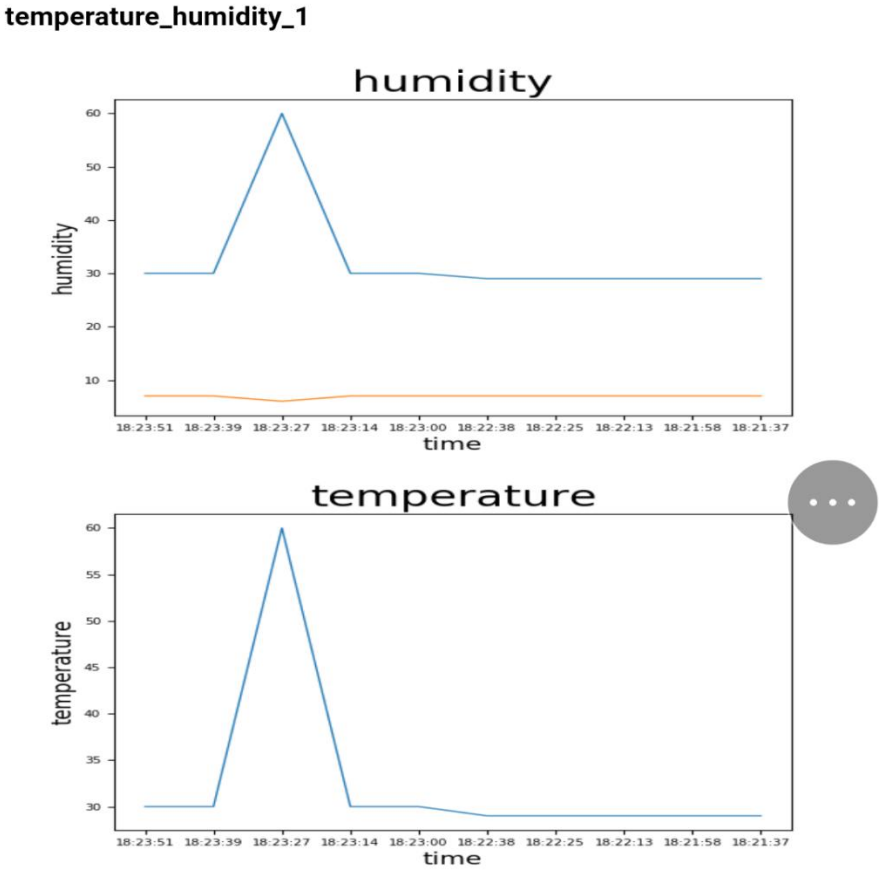
temperature_humidity_1
humidity
humidity
60
50
40
30
20
10
18:23:51 18:23:39 18:23:27 18:23:14 18:23:00 18:22:38 18:22:25 18:22:13 18:21:58 18:21:37
time
temperature
temperature
60
55
50
45
40
35
30
18:23:51 18:23:39 18:23:27 18:23:14 18:23:00 18:22:38 18:22:25 18:22:13 18:21:58 18:21:37
time

图 7.6　手机访问结果图

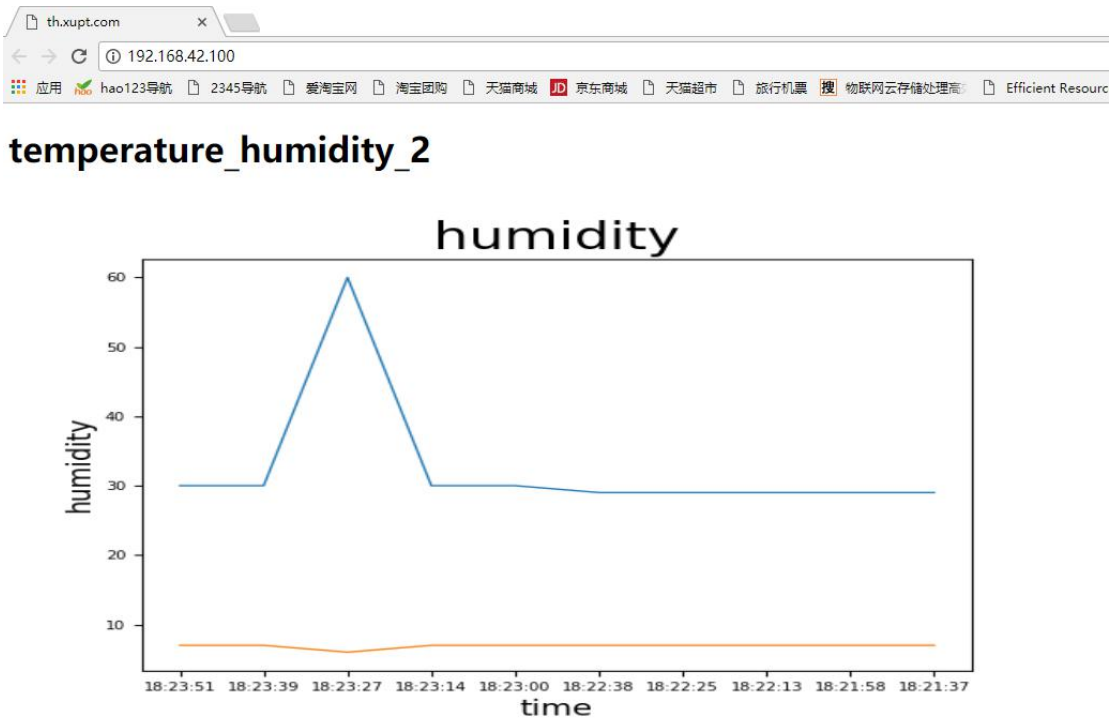

图 7.7　谷歌浏览器访问结果

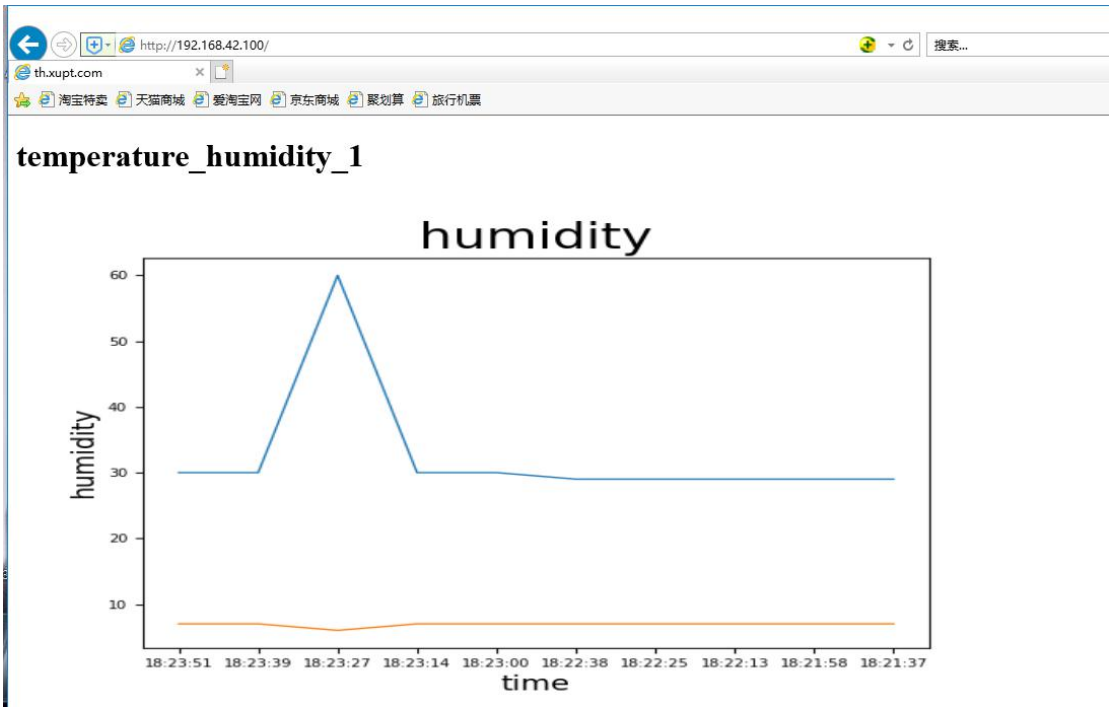

图 7.8　IE 浏览器访问结果

根据以上测试结果来看，该云平台架构能够满足中小物联网企业对于固定架构的要求且该架构的扩展也十分简单，当服务器由于访问量而达到架构上限时，只需要以 Server3 或者 Server4 为模版进行克隆机器并更改 Server1，Server2 的

keepalived 配置就可以完成横向扩展。由于本次压测属于内网压测所以其结果的真实度中的响应时间并不可靠，其响应时间主要是由网络环境决定的，但该架构支撑 500 至 1000 的 RPS 还是能够做到的，这也基本上接近一个中小物联网企业网站的极限。

结束语

本次毕业设计基于基于虚拟机模拟公有云服务器，实现对物联网云平台的架构设计。此次设计首先通过DHT11温湿度传感器获取环境数据，通过树莓派实现数据的远程上传，再通过虚拟机完成对数据库数据的调用和显示。最终实现用户可以在互联网下通过个人PC或者手机获取到所需信息。在设计的最后阶段对整体框架进行了压测测试。使其的可靠性的以验证，对于中小型的物联网企业来说，一套合理且高效的架构体系可以快速使其完成产业转型抢占市场份额，而现今人们意识形态的改变使得企业必须适应潮流否则必将被淘汰，所以对于物联网云平台的架构研究是很有必要的。

当然本次毕业设计中也存在着一些问题和瑕疵，首先由于没有使用真正的公有云服务器导致模拟环境和真实生产之间还存在一定的差异，其次现今的公有云厂商都有提供一部分收费的监控服务而在此次毕业设计中，并未涉及到过多的对与整体架构的监控。这也可能是这套架构体系中的不足之处。

总之随着科技的不断发展物联网和云服务之间的结合会不断加深，对与物联网的云品台研究是有价值和有意义的。

致　谢

首先非常感谢李老师和马老师在此次毕业设计中给予我的帮助，在此次的毕业设计也是在他们的悉心指导之下完成的。他们为我讲解了此次毕设题目大体的思路，为我点明设计的核心。从刚开始的毕业设计的选题，到后来设计所需的 相关资料的查找与搜索，再到最后的设计的进一步修改与完善，老师都给予我了很大的帮助。每次我的设计出现一些解决不了的问题的时候，都会在老师耐心的讲解中，我逐渐理清思路找到问题的关键，我的毕业设计才可以顺顺利利的进行下去。

在论文即将结尾的时候，我的毕业设计也即将完成。从刚开始的选题到如今毕业设计将要结束，有太多可敬的师长、同学、朋友给与我帮助，再次诚挚的感谢他们。同时也要感谢我的母校西安邮电大学对我四年的精心培养。

参考文献：

[1]李芳颂，刘晓华，王锋.云服务平台关键技术的研究[J].通讯世界，2015（2）：7-8. [2]王超，骆德汉，郑魏，等. 基于 STM32 的嵌入式智能家居无线网关设计[J].计算机技术与发展，2013，23（3）：241-244.

[3]温凯峰.基于云平台的场所智能控制及报警系统的设计[J].嘉应学院学报（自然科学），2015，33（11）：24-28.

[4]曾红，党盼盼.基于 W5500 的嵌入式系统以太网网关设计[J].网络安全技术与应用， 2015，（2）：36-37.

[5]杨博，陈新，袁建辉，等. 基于多元自感知的车辆只能服务系统设计[J].物联网技术，2016，6（7）：37-40.

[6]董清，洪歧.一种基于云平台的智能家居光控系统设计[J].物联网技术，2017，7（4）：88-90.

[7]王慧敏.基于物联网及云计算的旅游安全保障平台的设计与搭建[J].黑龙江科技信息，2013（3）：29.

[8] 张金淦, 切入企业级应用的 Ubuntu[J], 软件世界,2007,13.

[9] 黄媛,郭利朋,杨英茹,高欣娜,靳青,岳赵寒, 一种基于物联网技术的智能温室云服务平台设计方案,安徽农业科学，2015，31.

[10] A.M.Gorelik, Statements, data types and intrinsic procedures in the Fortran standards, ACM SIGPLAN Fortran Forum, 2015, Vol.33 (2), pp.8-20