河南理工大学

硕士学位论文

物联网终端网络管理研究与实现

申请人姓名：孟千胜

指导教师：杨立身

学位类别：工学硕士

专业名称：计算机应用

研究方向：计算机网络

河南理工大学计算机科学与技术学院

二〇一二年六月

河 南 理 工 大 学

学 位 论 文 原 创 性 声 明

本人郑重声明：所呈交的学位论文：物联网终端网络管理研究与实现，是我个人在导师指导下进行的研究工作及取得的研究成果。论文中除了特别加以标注和致谢的地方外，不包含任何其他个人或集体已经公开发表或撰写过的研究成果。其他同志对本研究的启发和所做的贡献均已在论文中作了明确的声明并表示了谢意。

本人愿意承担因本学位论文引发的一切相关责任。

学位论文作者签名：

年　月　日

河南理工大学

学位论文使用授权声明

本学位论文作者及导师完全了解河南理工大学有关保留、使用学位论文的规定，即：学校有权保留和向有关部门、机构或单位送交论文的复印件和电子版，允许论文被查阅和借阅，允许将本学位论文的全部或部分内容编入有关数据库进行检索和传播，允许采用任何方式公布论文内容，并可以采用影印、缩印、扫描或其他手段保存、汇编、出版本学位论文。

保密的学位论文在解密后适用本授权。

学位论文作者签名：　　**导师签名：**

年　月　日　　年　月　日

中图分类号：TP317.4　　密　　级：公开
UDC：620　　单位代码：10460

物联网终端网络管理研究与实现

Research and Implementation of the network management of Internet of Things

申请人姓名 孟千胜　　申请学位 硕士
学科专业 计算机应用　　研究方向 计算机网络
导　　师 杨立身　　职　　称 教授
提交日期 **2012.04**　　答辩日期 **2012.06**

河南理工大学

致谢

本文是在我的导师杨立身教授的悉心指导和严格要求下完成的。杨老师以他独特的视角、敏锐的观察为我选定以物联网终端网络管理为主题的研究方向。并在本研究进行的过程中，论文撰写的过程中，他给我详细而具体的指导。对于每一次的我的疑问和难题，他总是给我以明确而具体的指导，让我一次次重新拾起信心，能够将课题继续进行下去。在他指导我的课题研究、论文撰写的过程中也让我同时体验到了做事的认真态度、良好的思维习惯等对研究工作是重要性。从中我也学到了面对问题的思维方式、解决问题的方式方法。在此我对杨老师致以最真挚的感谢。感谢王建芳博士在嵌入式开发方面给予的指导和帮助。感谢研究室主任汤永利博士给予的帮助。感谢同一个研究室的老师、同学们营造一个良好的研究学习环境，以及他们在学习和研究中的帮助。本文在写作的过程中，得到宿舍好友庞小澎、刘延、朱大磊、刘坤、汪高磊等在生活和学习上的照顾和帮助，感谢他们。感谢同学们在学习和生活中的帮助以及在工作中的理解与支持。

摘要

物联网作为未来技术产业的推动力量，已经在各个大公司得到重视和长足的发展，然而作为技术和资金实力都不济的小公司，在物联网实现和终端管理等技术方面还是不能够自给自足。本文根据矿山信息化河南省高校工程技术研究中心在物联网终端实现和管理研究方面的需求，为了实现物联网并对其终端进行网络管理展开研究。

本课题首先提出了物联网终端模型，通过对物联网终端相关硬件和软件技术的分析与研究，从而设计出了本课题研究的物联网终端的软件和硬件结构并描述了对采集到的物联网数据的处理流程。为了支持双协议栈和 SNMP 的移植及物联网终端平台的扩展，选择了嵌入式 linux 做为嵌入式操作系统平台，通过对嵌入式 linux 内核的裁剪配置实现了物联网终端对 IPv6 协议栈的支持。把 NET-SNMP 开发包作为简单网络管理协议（SNMP）的开发平台，设计 MIB 文档并开发委托代理程序，从而实现把采集到的数据实时地存入到委托代理的 MIB 库中。通过构建的交叉编译环境将使用 NET-SNMP 开发实现的委托代理程序进行编译并移植到终端，从而完成物联网终端对 SNMP 的支持。实现了对物联网终端进行网络管理的目的。最终形成了一个实现对物联网终端进行网络管理的可行方案。

关键词：物联网;嵌入式技术;简单网络管理协议;代理;温度传感器;网络管理

Abstract

As the core technology of the future information industry, Internet of Things is already been seen and developed suffciently in several enterprise. But there are also some technologic deficiencies and problems during the Internet of Things terminal process of implementation and management in small companies. This thesis makes enough research on implementation and network management of Internet of Things based on the requirements of Mine of information technology in Henan Higher Engineering Research Center about terminal implementation and management research of IoT (Internet of Things).

The study first proposed the Internet of Things terminal model,then designs hardware and software structure of an Internet of Things terminal and makes abundant analysis on related technologies, including the selection of software and hardware on Internet of Things and its management，and describes the processes the collected data of Internet of Things. To support migration of dual protocol stack and SNMP (simple network management protocol) and expansion of Internet of things Terminal platform, Linux system is selected as embedded operating system platform to make the Internet of Things terminal support IPv6 protocol stack according to cutting and configuration of Linux core. Simultaneously, NET-SNMP Development Kit is as development platform of SNMP to design MIB documents and develop the Proxy Agent programming, in order to achieve the data collected by polling real-time credited to the Proxy Agent's MIB; Build cross-compilation envirment, and transplate SNMP Proxy Agent to the terminal, thus completing the Internet of Things SNMP support.Finally,achieve the purpose of network management terminal of IoT(Internet of Things) .Eventually, form an implementation of feasible options for network management of Internet of Things.

Keyword: Internet of Things; Embedded technology; Simple Network Management Protocol; agent; temperature sensor; network management

目录

1 绪论

1.1 选题依据

在物联网的概念提出之后，随着物联网关键技术发展的日益成熟，物联网的内涵也随之悄然发生了变化。同时，物联网技术也随之在国内外成为热潮，大家普遍认为物联网将是一场新的“工业革命”，它作为未来技术产业的推动力量已经在现实生活中崭露头角。我国敏锐地捕捉到了这一科技发展的良好时机，将物联网作为战略性新兴产业加以重点关注与推进。国家领导人也对物联网技术的发展表示了高度的重视，2009 年 8 月，温家宝总理在江苏无锡考察时提出了“感知中国”，由此推动了物联网概念和技术在国内的重视。在计算机、互联网及移动通信等技术引发了信息产业发展的浪潮之后，物联网成为这个产业的核心领域。温家宝总理于 2009 年 11 月份进行南京考察时则表示：当前，流通行业要大力运用网络技术，尤其是物联网技术，尽早实现流通现代化[1] 。温总理在 2010 年 3 月份作的《政府工作报告》中更是将“加快物联网的研发应用”明确纳入重点振兴产业。

1999 年物联网的定义被首次提出来，它具体的定义是：把所有物品通过射频识别等信息传感设备与互联网连接起来，实现智能化识别和管理[2] 。“物联网”作为一个新的概念，它打破了之前的传统思路，开拓了一个全新的技术领域。传统的思路就是物理上分开物理基础设施与 IT 基础设施。一边是机场、公路、建筑物等物理的基础设备；在另一边则是数据中心、个人电脑、宽带等 IT 设备。但是在“物联网”时代，芯片、宽带将跟钢筋混凝土、电缆被整合，成为一体的基础设施。在此意义上的“基础设施”更像是一块新的地球工地，整个世界，无论是经济管理、生产运行还是社会管理，甚至是个人生活的运转都是在它上面进行的。这样就能够达到智能化地提高资源利用率以及生产力水平，并改善人与自然之间的相互关系。

物联网拥有丰富的应用和数量庞大的节点。这在商业上带来了巨大的经济潜力，同时也在技术实现上带来了新的挑战。

首先，众多的节点连接构成的物联网节点之间的通信问题，这是与寻址分不开的。现在，物联网可采用的寻址方式有两种：一种是采用基于 E.164 电话号码编址的寻址方式。然而大多数物联网应用都把 TCP/IP 协议作为网络通信协议。

因此这种方式也必然要求电话号码和IP地址之间进行相互转换。另外，鉴于E.164编址体系自身包含的地址数量较少，它根本无法满足物联网节点对海量地址需求。另一种方式则是通过利用 IPv4 地址寻址系统，对物联网节点进行寻址。然而如今的 IPv4 的地址已经分配殆尽，物联网对网络地址的海量需求使得如今的IPv4 的地址空间根本无法满足。另外，考虑到物联网对海量地址的需求，在海量地址的分配上已经无法再使用人工分配了，这也同时提出了改变地址分配方式的要求，如果使用传统的 DHCP 的分配方式，则会对现有互联网中的 DHCP 服务器的可靠性和性能要求极其苛刻，同时会使得现有 DHCP 服务器在性能上不堪重负，成为物联网的大规模应用和发展的另一个瓶颈[3] 。

其次，目前以 IPv4 为基础的互联网的移动性不足成为满足物联网对移动能力要求的一个瓶颈。IPv4 协议在最初的设计中并没有对移动性带来的路由问题进行充分的考虑，也就是是说，它没有考虑到当一个节点移动出它原有的网络之后，如何才保证其他节点仍然能够与这个节点互相通信的问题。虽然在后来的设计中 IETF 提出了 MIPv4 机制用以对网络节点的移动进行支持。但是这个机制的引入同时造成了著名的三角路由问题：MIPv4 机制在解决少量节点的移动问题上，所引起的网络资源损耗较小，效果也比较好，然而在应对大量节点的移动，尤其是物联网中所特有的节点群的移动和节点层的移动上，则会迅速耗尽网络资源，从而造成网络的瘫痪[3] 。

再次，物联网节点在信息安全性的需求上更加迫切，在可靠性方面也需要进行新的考量。下一代互联网协议——IPv6 将是替代 IPv4 成为物联网的网络层的基础和核心技术。

1.2 国内外研究现状

2005 年 ITU 在突尼斯举行的信息社会世界峰会(WSIS)上正式确定了“物联网”的概念，并随后发布了《ITU Internet reports 2005———the Internet of things》。这个报告的推出，使得物联网的发展在全球范围内得到了重视。同时，物联网被视为在信息领域的发展和变革的重大机遇。

我国在物联网方面的研究起步相对来说较早。1999 年，中国科学院已经启动了传感器网络技术的研究，在传感器端机和移动基站等方面取得了重大的发展，同时在无线智能传感器网络通信技术和微型传感器也取得了重大突破。

发达国家也十分重视物联网相关技术和产业的发展。欧盟执委会在 2009 年

6 月 18 日发表了《Internet of things——an action plan for Europe》。其中前瞻性地描述了物联网发展的美好前景。并在世界范围内首次对物联网的发展和管理设想做了系统描述。同年 10 月 13 日，韩国通信委员会通过了《物联网基础设施构建基本规划》。其中制定了“通过构建世界最先进的物联网基础实施，打造未来广播通信融合领域超一流信息通信技术强国”的目标。还为物联网的发展确定了构建物联网基础设施、发展物联网服务、研发物联网技术、营造物联网扩散环境四大发展领域以及 12 项具体课题[4] 。

我国已经和美国、韩国、德国一起，成为物联网国际标准制定的主导国家。自 2009 年以来，无锡市已经联合美国有关机构发起了物联网国际标准的制定工作。物联网相关产品现在已经在公共安全、民航、交通、电网、环境监测等许多行业得到了初步应用。我国的物联网产品也已经远销美国、加拿大等国。

1.2.1 物联网定义探讨

物联网被视为在信息领域发展和变革的重大机遇。然而从整体上说，不管是国内还是国外，对物联网技术的研究和物联网产业的发展都仍然处在起步阶段，不同领域的专家学者研究的物联网技术起点各异，他们对物联网的定义也不能达成一致，对物联网的系统模型、体系架构以及关键技术的界定仍然不够清晰。下面讨论目前关于物联网的几个主流的定义，并对三层框架结构及相关技术做一个简述[5] 。

“物联网”（Internet of Things）这个词最初来源于美国麻省理工学院(MIT)是国内外业界所普遍认可的。它是指在 1999 年 MIT 建立的自动识别中心(Auto-IDLabs)提出的网络无线射频识别(RFID)系统：即把所有物品通过射频识别等信息传感设备与互联网连接起来，实现智能化识别和管理。最早的物联网是在物流行业智能管理需求的背景下提出的，用射频识别技术代替条码识别技术，从而实现对物流系统进行智能化管理的目的[6] 。然而，随着物联网技术和应用的进一步发展，物联网的内涵也悄然发生了很大的变化。

在 2005 年国际电信联盟（ITU）在突尼斯举行的信息社会世界峰会(WSIS)上正式确定了“物联网”的概念，并随后发布了《ITU Internet reports 2005—the Internet ofthings》以及欧盟 2008 年发布的《the European Technology Platform on Smart Systems Integration，EPoSS IoT 2020》报告中，物联网的定义和范围就已经发生了较大变化，物联网不再仅仅指基于 RFID 技术的。它的覆盖范围也有了

较大的拓展。

随着相关学科的发展，信息领域科研工作者分别从不同的方面对物联网进行了颇为深入的研究，物联网的概念范畴也随之发生了深刻的变化。然而时至今日仍然没有提出一个权威的物联网定义，对它进行完整并精确的描述。

由于不同领域的研究者对物联网思考所基于的起点各异，对物联网的定义上也侧重于不同的方面，在短期内还没有就此达成共识。几个具有代表性的物联网定义如下。

定义 1：物联网是未来网络的整合部分，它是以标准、互通的通信协议为基础，具有自我配置能力的全球性动态网络设施。在这个网络中，所有实质和虚拟的物品都有特定的编码和物理特性，通过智能界面无缝链接，实现信息共享[7] 。

定义 2：由具有标识、虚拟个性的物体/对象所组成的网络，这些标识和个性运行在智能空间，使用智慧的接口与用户、社会和环境的上下文进行连接和通信[8] 。

定义 3：通过射频识别（RFID）、红外感应器、全球定位系统、激光扫描器等信息传感设备，按约定的协议，把任何物品与互联网连接起来，进行信息交换和通信，以实现智能化识别、定位、跟踪、监控和管理的一种网络。它是在互联网基础上延伸和扩展的网络[9] 。

目前最普遍引用的物联网定义是“定义 3”， 然而现在看来，这个定义显然即不全面，也不严谨，而且存在错误。例如，物联网可以存在于内网和专网之中，而且现在大部分物联网还只是存在于内网和专网例如智能交通、智能电网等各种物联网的应用，“把任何物品与互联网连接起来”，这个说法现在看来显然是不准确的，同时这也明显地显现出了这个定义的不全面和不严谨。

国家物联网首席科学家刘海涛说：物联网就是以感知为目的的物物互联的综合的信息系统，简单说八个字形容，物物，互联，感知，系列。从学术角度来讲，它叫传感网，物联网是大众化的说法。中国对物联网的研究都是指以感知为核心，它不是简单的物物互联，而是感知，更多的是一种群体效应的社会化的感知。

究竟是传感器技术还是网络技术在物联网技术中起主导地位，不同的人看法不尽相同，比如有人认为感知是物联网的基石，这代表了认为传感器是物联网技术的核心一派的观点；也有人认为物联网是互联网的延伸，这个观点代表了认为网络应在物联网中占主导地位的一派的观点；无论是感知还是网络，虽然两派的定义不同，但是对于物联网却都有一些共同的认识，那就是物联网的三层架构。

即：下面一层叫感知层，中间层是网络层，上面一层是应用层，这三层分别对应着物联网的三大要素即：信息采集，信息传输，信息处理。因此在物联网里面，只要记住它的三大要素，这三大要素构成了物联网。

1.2.2 物联网三层框架结构

如图 1-1 所示物联网的体系架构，物联网可分为三层即：感知层、网络层、应用层。感知层是物联网的基础，是信息收集的原点，是信息的来源，它利用传感器、RFID 等技术随时随地采集物体的信息。网络层是信息可靠传输的保证、是应用层和感知层的桥梁，是物联网的支撑，它通过互联网、电信网、广播电视网的融合将物体的信息实时准确的传输出去。应用层用于实现物联网的应用，即对采集到的数据信息进行智能处理，主要使用模糊识别、云计算等智能数据处理技术，对物联网收集的海量信息与数据进行处理，从而实现智能化控制。总之，应用是目的，感知是基础，网络是信息传输的平台。

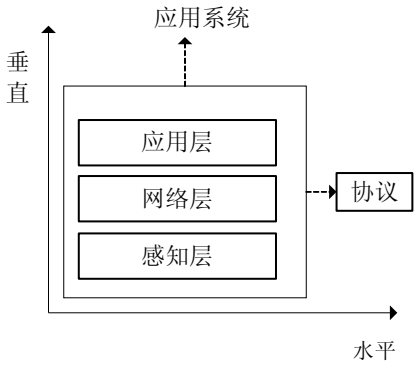

图 1-1 物联网的体系架构

Fig.1-1 Internet of Things architecture

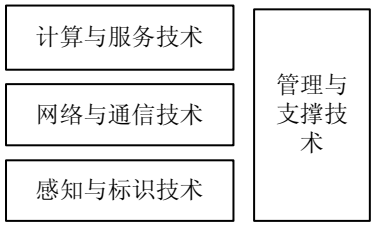

图 1-2 物联网的技术体系模型

Fig.1-2 Internet of Things technology model

目前，RFID、普适服务、云计算及传感器等领域是我国物联网技术所主要的关注的热点领域。然而物联网技术所涉及的领域很多，这些技术在其所属的行

业常常具有不同的应用需求和技术形态。参照如图 1-1 所示的物联网结构，并对物联网所涉及的相关技术进行总结和归纳，可以形成如图 1-2 所示的物联网技术体系模型。在这个技术体系中，物联网的感知层技术是感知与标识技术，网络层技术是网络与通信技术，应用层技术是计算与服务技术，另外还有不能归属于单独的任何一层的管理与支撑技术。

1.2.3 感知层及其技术

感知层由智能传感器节点和接入网关组成，接入网关收集智能节点感知到的信息，这些信息通过网络层传输到应用层进行数据的处理。数据采集与信息感知主要是指对物理世界中发生的物理事件的感知和相关物理数据的采集，如各类物理量、标识、音频、视频数据。感知层技术是物联网发展和应用信息来源的基础。传感和控制技术，短距离无线通讯技术（WIFI），RFID 技术等这些发展成熟度差异性相当大的技术就是感知层相关的主要技术。

上述的这些感知和标识技术是物联网技术的基础，负责采集数据和感知信息，实现对物理世界信息的感知和识别。

1)传感技术利用传感器或者传感器网络节点间的协作，感知并采集传感器覆盖区域中被感知对象的可感知信息。传感器技术发展程度有赖于工艺设备、计测技术、敏感材料及敏感机理的发展程度，因此它对基础技术和综合技术等相关技术要求都特别高。现在，传感器在被检测量的种类、精度、稳定性、可靠性、低成本、低功耗等多个方面的技术都有很大发展，然而这些技术仍然没有达到大规模应用的水平，这也成为物联网产业化发展与大规模应用的重要瓶颈之一[10] [11] [12] 。在本课题中，选择发展比较成熟的温度传感器作为感知层进行开发。

2)识别技术包括位置识别、地理识别和物体识别等。对现实世界的物理信息识别是实现全面感知的基础。对象标识体系是物联网技术的一个重要技术点，物联网的对象标识体系是以 RFID 标识、二维码等标识技术为基础的。从应用需求的角度上说，识别技术首先必须解决的问题是对对象的全局标识，这需要更深入的研究和完善物联网的标准化物体标识体系，进一步融合并适当的支持现有的和未来的各种传感器和标识方案。

1.2.4 网络层及其技术

物联网的网络层是建立在现有的互联网和移动通讯网的基础上的，互联网和

移动通讯网是物联网信息传递和服务支撑的基础设施，物联网通过各种连网设备接入到互联网及移动通讯网中。网络要能够把感知到的信息高安全性、无障碍地进行高可靠传输，就要移动通信技术、互联网技术同传感器网络的相互融合，实现信息的互联功能。此外，网络层还应具有包括信息的存储与查询、网络管理在内的各种辅助功能。

物联网网络技术主要包含骨干传输、泛在接入等多个方面的相关技术。其中，以互联网协议 IPv6 为核心的下一代网络技术，为物联网的发展创造了良好的网络基础条件。固定及无线网技术、移动网技术、Ad-hoc 网技术、自治计算与连网技术等都需要更深入的研究，因为以传感器网络为代表的末梢网络在规模化应用后，面临着与骨干网络的接入问题，并且其网络技术需要与骨干网络进行充分协同，这些都将面临着新的技术难题需要解决[13] 。

作为实现以数据为中心的物联网的核心技术，网络层中对感知数据处理与管理的技术包括基于感知数据的决策和行为的技术与理论、传感网数据的存储、挖掘、查询、分析及理解。作为海量感知数据的存储、分析平台的云计算平台是物联网网络层的重要组成部分。同时云计算也是应用层实现众多应用的基础平台。

物联网需要综合各种有线及无线通信技术，其中物联网的研究重点是近距离无线通信技术。工业科学医疗(ISM)频段是全世界都可通用的免许可证的 2.4 GH 频段，由于物联网终端一般使用它进行通信，这个频段内包括大量的物联网设备以及现有的无线保真(WiFi)、超宽带(UWB)、ZigBee、蓝牙等设备，在物联网大规模应用时这个频谱空间将极其拥挤，频谱空间也将成为物联网的实际大规模应用的制约因素之一。为提升频谱资源的利用率，保证异种物联网的共存，让更多物联网业务能实现空间并存，并实现其互联互通互操作，需要切实提高频谱保障能力以应对物联网的规模化应用。

1.2.5 应用层及其技术

应用层是物联网发展的目的，物联网采用软件开发、智能控制技术为用户提供多种多样的服务和应用。各种家庭应用与行业应用的开发必定会促使物联网的普及，同时也会给整个物联网产业链带来利润。目前已经有不少基于物联网的应用，比如使用某一种传感器感应到某个物体触发的信息，然后按已定的智能处理设定通过网络完成一系列动作。当你早上拿车钥匙出门上班，在电脑旁待命的感应器检测到之后就会通过网络自动发出一系列的命令，从而触发一连串的事件：

使用短信或者喇叭自动报出今天的天气；在电脑上显示快捷通畅的开车路径并估算路上所花时间；同时利用即时聊天工具或者短信通知你的同事你即将到达。又如已经投入试点运营的高速公路不停车收费系统，基于 RFID 的手机钱包付费应用等。

物联网的核心支撑是对海量感知信息的计算和处理。物联网的最终价值主要体现在它所能够提供的服务和应用上。物联网应用层将感知数据经过分析和处理后，能够为用户提供丰富的服务。

应用层主要包括应用服务子层和应用支撑平台子层。其中应用支撑平台子层用于支撑跨行业、跨应用、跨系统之间的信息协同、共享、互通的功能。应用支撑平台子层中海量感知信息计算与处理技术是物联网应用大规模发展后，所面临的重大挑战之一。为了攻克物联网“云计算”中的虚拟化、网格计算、服务化和智能化技术，需要研究海量信息的数据融合、高效存储、语义集成、并行处理、知识发现和数据挖掘等关键技术，其核心是为海量信息的高效利用提供支撑。这需要采用云计算技术实现信息存储资源和计算能力的分布式共享。应用服务子层包括智能交通、智能医疗、智能家居、智能物流、智能电力等行业应用。物联网的应用服务可分为监控型（污染监控），查询型（智能检索），控制型（智能交通），扫描型（高速公路不停车收费）等。

物联网的发展应以应用为核心，物联网服务的模式会将服务的内涵进行革命性的扩展，不断出现的新型应用必将对物联网的服务模式与应用开发进行一次又一次的巨大挑战，如果仍然沿用传统的技术路线必将会束缚物联网应用的创新。为了适应未来应用环境变化和服务模式变化，需要面向物联网在典型行业中的具体应用需求，研究针对不同应用需求的规范化、标准化服务体系结构及其应用支撑环境和面向服务的计算技术等，提炼出行业普遍存在或共同需要的核心共性支撑技术。

1.2.6 三层共用的技术

随着物联网应用规模的大幅度扩大、服务质量要求的进一步提高和承载业务的多元化以及影响网络正常运行因素的增多，一些不能独立于物联网三层中的任何一层的管理和控制技术是保障物联网“可运行-可管理-可控制”目标的实现的关键，这些技术包括测量分析、网络管理和安全保障等三个方面。

1)可测性是网络管理研究中的基本问题，解决网络可知性问题的基本方法是

测量。随着网络复杂性的提高以及物联网新型业务的不断涌现，研究高效的物联网测量分析技术的必要性更加迫切，并同时需要建立面向服务的物联网测量机制与方法。

2)物联网具有自治、开放、多样的自然特性，它与网络进行网络管理的基本需求存在着突出矛盾，为保证网络系统正常高效的运行需要研究新的物联网管理模型与技术。

3)安全是以网络为基础的各种系统正常运行的重要基础之一，然而物联网的开放性、包容性和匿名性决定了其不可避免地存在着各种信息安全隐患。需要研究物联网安全技术，以满足物联网信息机密性、真实性、完整性的要求，同时还需解决好物联网中的用户隐私保护与信任管理问题。

1.2.7 物联网网络管理

由上文可知，物联网并不是新的事物,它可以看作由传感器或传感器网络接入互联网构成的网络，当然也有由传感器网络自身组成的专网。但是总体上来说，由于物联网具有许多新的特点，从而导致了物联网对于其网络的管理有新的要求。因此物联网的网络管理还是一个新的工作。到现在，还少有专门针对物联网网络管理的研究。在现实应用中，有些对物联网远程管理使用的是 Telnet，管理主要使用人工进行，这对于大规模的物联网节点的管理是很不合适的，Telnet 的使用有一个很大的问题，就是安全性，它没有强健的安全机制，根本不符合物联网对安全性的要求；而有些干脆不进行管理，也就是说对物联网的管理缺失。

基于 TCP/IP 网络的 SNMP 简单网络管理协议以其简单实用而被广泛应用于现有的计算机网络中，是应用最广泛的 TCP/IP 网络管理框架，已经成为网络管理的事实上的标准。

SNMP 的网络管理模型包括以下关键元素：管理站、代理者、管理信息库和网络管理协议，其中管理站一般是一个分立的设备，也可以利用共享系统实现，管理站被作为网络管理员与网络管理系统的接口；代理者装备了 SNMP 的平台，如主机、网桥、路由器、集线器以及支持 SNMP 的物联网终端均可作为代理者工作，代理者对来自管理站的信息请求和动作请求进行应答，并随机地为管理站报告一些重要地意外事件；MIB(管理信息库)是设在代理者处的管理对象的集合，管理站通过读取 MIB 中对象的值来进行网络监控。管理站和代理者之间采用 SNMP 网络管理协议通信。在现有的计算机网络中无论集中式网络管理、分层式

网络管理以及基于 CORBA 的网络管理和基于 Web 的网络管理等分布式网络管理技术都是基于 SNMP 的网络管理技术。

由于 SNMP 的开放性、设计简单、能够扩展等优点，在对物联网进行网络管理时，也同样选择 SNMP 的网络管理模型，对物联网终端进行网络管理，这样就可以同时借鉴现有网络的管理模型，为以后对物联网管理的进一步完善提供支持。

1.3 研究内容及论文结构

本课题根据矿山信息化河南省高校工程技术研究中心针对物联网研究方面的需求确定的课题。将以温度传感器为起点，研究物联网的网络模块，利用嵌入式技术，将简单网络管理协议移植到嵌入式开发系统中，为物联网节点的网络管理提供支持。为传感器节点接入计算机网络并进行网络管理提供解决方案。并对方案进行应用分析、设计、研究和总结。

论文主要分为 4 章，主要内容安排如下：

第 1 章：主要介绍本课题研究的背景、研究现状、本课题主要的研究内容并分析了物联网整体架构、物联网网络管理现状及相关技术。

第 2 章：提出了本课题的物联网终端模型，并根据研究的需要，分析并研究相关的硬件技术和软件技术，为课题的物联网硬件结构和软件结构提出提供理论支持。

第 3 章：为本课题研究提出了物联网终端的硬件结构和软件结构，并描述了该物联网终端的数据处理流程，然后描述了本课题研究内容的实现过程。

第 4 章：对研究的内容进行总结，归纳出研究的收获并找出不足，提出未来研究的方向。

2 物联网终端模型及相关硬、软件技术研究与分析

为了研究物联网终端，本章首先提出了物联网终端模型。根据实现物联网终端及其网络管理的需要，必须选择相应的硬件，确定物联网终端的主要组成，选择相应的软件，实现网络管理的功能，这里就需要根据需求，分析相关技术，并做好相应的研究工作。

本章通过对物联网终端使用的硬件（ARM、智能信息采集设备——温度传感器）技术和软件（NET-SNMP、嵌入式 linux）技术等相关技术进行分析和研究，为终端硬件架构设计和软件架构设计提供理论支持。

2.1 物联网终端模型

物联网终端模型结构如图 2-1 所示。

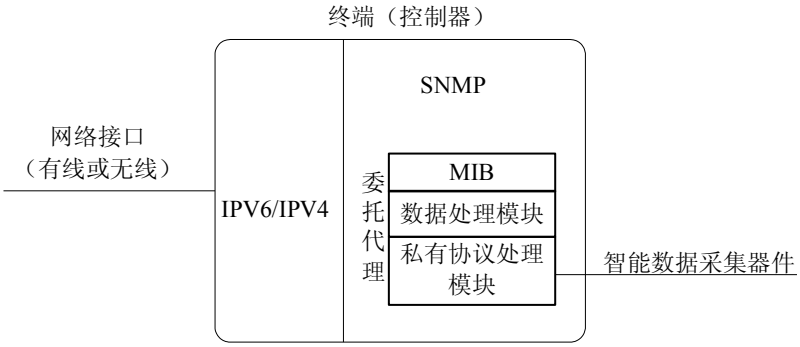

图 2-1 物联网终端模型结构

Fig.2-1 Terminal of IoT model structure

物联网终端模型中，终端通过网络接口与 internet 连接，这个终端需要更好的与计算机网络连接，有必要支持 IPv6/IPv4 双协议栈。为了达到网络管理的目的，它需要支持 SNMP，使用 SNMP 的委托代理网管模型，实现网络管理的目的。采用委托代理处理智能数据采集设备采集的数据，将其保存到 MIB 库中。委托代理中通过私有协议处理模块与智能数据采集设备交换数据，通过数据处理模块更新 MIB 库数据，为管理器提供最新数据。

需要说明的是：智能数据采集设备采集到的数据可能是模拟信号，也可能是数字信号，而终端（控制器）只能够识别电子信号，对于采集到的模拟信号需要将其转换成数字信号，在这里我们假定其采集到的信号是数字信号，能够被终端中程序直接进行处理。

2.2 硬件技术分析

2.2.1 ARM 及其技术分析

目前嵌入式处理器常见的有 ARM、PowerPC、MIPS、Motorola 68K、ColdFire(冷火)等，但 ARM 占据了绝对主流。ARM 架构，过去称作进阶精简指令集机器，是一个 32 位精简指令集（RISC）处理器架构，它在许多嵌入式系统设计中被广泛应用[14] 。由于移动通讯领域对节能的要求，ARM 处理器非常适合于应用在这个领域，这也符合 ARM 处理器的主要设计目标：低成本、高效能、低耗电的要求。

ARM 的设计是 Acorn 电脑公司（Acorn Computers Ltd）于 1983 年开始的开发计划。这个团队于 1985 年时开发出了 ARM1 Sample 版。次年首颗“真正”的产能型 ARM2 量产。Acorn 的 ARM 处理器作为低成本 PC 的第一款 RISC 处理器在 1987 年产出。20 世纪 80 年代晚期，苹果电脑开始与 Acorn 电脑合作开发新版的 ARM 核心。紧接着，Acorn 电脑在 1990 年将设计团队另组成一家名为 Advanced RISC Machines Ltd.的新公司。因此 ARM 有时候称作 Advanced RISC Machine 而不是 Acorn RISC Machine。

ARM 公司自 1990 年正式成立以来，在 32 位 RISC (Reduced Instruction Set Computer)CPU 开发领域不断取得突破，其结构已经从 V3 发展到 V6。自从 1991 年 ARM 推出第一款嵌入式 RISC 核心（即 ARM6 解决方案）以来 ARM 已推出多达十几种的核心结构。其设计的芯核具有成本低、功耗低等显著的优点，这使它在 32 位嵌入式应用领域获得了巨大的成功，目前 ARM 是占全世界最多数的 32 位系统架构，据统计，ARM 家族占了所有 32 位嵌入式处理器产品市场 75%以上的份额。在低成本、低功耗的嵌入式应用领域确立了市场领导地位的 ARM 处理器可以在很多消费性电子产品上找到踪迹，例如：计算机、多媒体播放器、PDA、移动电话、掌上型电子游戏等可携式装置；桌上型路由器、硬盘等电脑外设；甚至在导弹的弹载计算机等军用设施中都有它的存在。此外，还有一些基于 ARM 设计的派生产品。

ARM 公司将处理器架构授权给有兴趣的厂家，而自己从不涉足芯片的生产销售。也就是说，它本身并不制造或出售 CPU。ARM 的经营模式在于出售其半导体知识产权核心（IP core），它一直以半导体知识产权提供者的身份向各大半导体制造商出售其知识产权，获得授权的厂家按照处理器内核架构设计制作出建

构于此内核的微控制器和中央处理器。它最成功的实际案例是 ARM7TDMI，这套内核几乎卖出了数亿套装置。

多家半导体公司持有 ARM 的 IP(Intelligence Property)授权：Atmel，富士通，Broadcom，Cirrus Logic，Freescale，英特尔，IBM，NVIDIA，英飞凌，任天堂，恩智浦半导体，OKI 电气工业，三星电子，Sharp，STMicroelectronics，德州仪器和 VLSI 等许多这些公司均拥有各个不同形式的 ARM 授权。在此工业领域，人所共知的是 ARM 是最昂贵的 CPU 内核之一，仅包含一个基本的 ARM 内核的单一客户产品就可能需索取一次高达 20 万美金的授权费用，然而如果关系到大量的架构的修改，其费用就可能动辄千万美元以上。

目前比较流行的 ARM 芯核有 ARM7TDMI，StrongARM ARM720T，ARM9TDMI，ARM922T，ARM940T，RM946T，ARM966T，ARM10TDM1，ARM1176JZF 等。自 V5 以后，ARM 公司提供 Piccolo DSP 的芯核给芯片设计者，用于设计 ARM+DSP 的 SOC(System On Chip)结构的芯片。另外，ARM 芯片也获得了许多知名的实时操作系统产品供应商的支持，如：Windows CE、Linux、pSOS、VxWorks Mucleus、EPOC、uCOS、BeOS 等。

2.2.1.1 设计特点

讲求精简又快速的设计方式，整体电路化却又不采用微码，就像早期使用在 Acorn 微电脑的 8 位 6502 处理器。

ARM 架构包含了下述 RISC 特性：

1、读取／储存架构

2、不支持地址不对齐内存存取（ARMv6 内核现已支持）

3、正交指令集（任意存取指令可以任意的寻址方式存取数据 Orthogonal instruction set）

4、大量的 16×32-bit 寄存器阵列（register file）

5、固定的 32bits 操作码（opcode）长度，降低编码数量所产生的耗费，减轻解码和流水线化的负担。

6、大多均为一个 CPU 周期执行。

为了补强这种简单的设计方式，相较于同时期的处理器如 Intel 80286 和 Motorola 68020，ARM 还多加了一些特殊设计：

1、大部分指令可以条件式地执行，降低在分支时产生的负重，弥补分支预

测器（branch predictor）的不足。

2、算数指令只会在要求时更改条件编码（condition code）。

3、32-bit 筒型位移器（barrel shifter）可用来执行大部分的算数指令和寻址计算而不会损失效能。

4、强大的索引寻址模式（addressing mode）。

5、精简但快速的双优先级中断子系统，具有可切换的暂存器组。

ARM 处理器还有一些在其他 RISC 的架构所不常见到的特色，例如 PC-相对寻址以及前递加或后递加的寻址模式。

2.2.1.2 芯片选择

随着我国在嵌入式应用领域的快速发展，ARM 芯片开始获得了大家的重视和应用。但是，ARM 千变万化的内部功能模块的组合配置，多达十几种的芯核结构和 70 多家芯片生产厂家的不同芯片，这些都给开发人员在芯片选择时带来了不少的困难。对此，需要分析芯片的特点：

1、MMU

并不是所有的芯片都具有 MMU，如果您为了减少软件开发时间而使用 WinCE 或 Linux 等操作系统，有必要选择带有 MMU(memory management unit)功能的 ARM 芯片[15] [16] 。带有 MMU 功能的芯片有 ARM720T、StrongARM、ARM920T、ARM922T、ARM946T 等。而 ARM7TDMI 没有 MMU，也不支持 Windows CE 和大部分的 Linux，但像 uCLinux 等少数几种 Linux 不支持 MMU 功能。

2、系统时钟控制器

不同的芯片对时钟处理也有所不同，有的芯片只有一个主时钟频率，这样的芯片不能保证 UART 和音频时钟的准确性；而有的芯片的内部时钟控制器能够分别为 CPU 内核、USB、音频、DSP 和 UART 等不同部件提供不同频率的时钟[17] [18] [19] 。ARM 芯片的速度由系统时钟决定。ARM7 的速度为 0.9MIPS/MHz，ARM7 芯片系统主时钟为 20MHz-133MHz，ARM9 的速度为 1MIPS/MHz，ARM9 的系统主时钟为 100MHz-233MHz，ARM10 最高可以达到 700MHz。

3、内部存储器容量

根据存储器的容量要求可以做出不同的选择，在对存储器容量要求不大时，带有内置存储器的 ARM 芯片是性价比较好的选择。

4、USB 接口

许多 ARM 芯片有内置的 USB 控制器，甚至有些芯片有 USB Host、USB Slave 控制器。

5、GPIO（General purpose input output）引脚数量

在某些芯片供应商提供的说明书中，往往说明的是最大的通用目的输入输出（GPIO）引脚数量，但是其中有许多是跟地址线、数据线、串口线等其它功能的引脚复用的。在系统设计时需要自己计算出可以实际使用的通用目的输入输出引脚数量。

6、中断控制器

由于 ARM 设计时对精简的要求，它的内核结构中仅提供两个中断向量：快速中断(FIQ)与标准中断(IRQ)。但是这对于一些应用设计来说，显然是不够的。于是半导体厂家在设计芯片时为了支持诸如串行口、外部中断、时钟中断等硬件中断而加入了自己不同的中断控制器。合理的外部中断设计可以很大程度的减少任务调度的工作量，因此在选择芯片时需要考虑外部中断控制[19] 。

7、IIS(Integrate Interface of Sound)接口

对于设计音频产品的应用时，IIS（集成音频接口）总线接口是必需的。

8、nWAIT 信号

不是每个 ARM 芯片都会提供外部总线速度控制（nWAIT）信号引脚，这个信号跟成本低廉的 GAL 芯片一起就能够可以实现符合 PCMCIA 标准的 WLAN 卡或 Bluetooth 卡的接口，而不用外加价格不菲的 PCMCIA 专用控制芯片。当扩展外部 DSP 协处理器时，需要这个信号。

9、RTC (Real Time Clock)

不同的 ARM 芯片以不同的方式提供实时时钟功能。

10、LCD 控制器

有些 ARM 芯片提供内置的 LCD 控制器，甚至内置 64K 彩色 TFT LCD 控制器。在进行设计 PDA、手持式显示记录设备等时，应当选用内置 LCD 控制器的 ARM 芯片。

11、PWM 输出

在电机控制或语音输出等场合选择带有 2~8 路 PWM 输出的 ARM 芯片。

12、ADC 和 DAC

在进行电池检测、触摸屏以及温度监测等应用时选择内置 2~8 通道 8~12 位

通用 ADC 的 ARM 芯片。

13、扩展总线

除了某些特殊应用的 ARM 芯片没有外部扩展功能外，大部分的 ARM 芯片都具有外部 SDRAM 和 SRAM 扩展接口。然而不同的 ARM 芯片能够扩展的片选线数量（芯片数量）有所不同，外部数据总线也有 8 位、16 位或 32 位几种。

14、UART 和 IrDA

所有 ARM 芯片都具有 1 到 2 个 UART 接口，可以用于 Angel 调试或者用于同 PC 机进行通讯。一般情况下 ARM 芯片通讯波特率为 115200bps，只有少数芯片的 UART 通讯波特率可以达到 920Kbps。这类 ARM 芯片是专为蓝牙技术的应用而设计的。

15、内置 FPGA

在开发通讯等领域的应用时，内置有 FPGA 的 ARM 芯片是首先。

16、看门狗与时钟计数器

通常，ARM 芯片都具有一个看门狗计数器以及 2 到 4 个 16 位或者 32 位的时钟计数器。

17、电源管理功能

ARM 芯片的耗电量与工作频率成正比，ARM 芯片都有低功耗模式、睡眠模式、关闭模式。

18、DMA(Direct Memory Access)控制器

为了减少数据交换时对 CPU 资源的占用，有些 ARM 芯片内部集成有 DMA 能够跟硬盘等外设高速地交换数据。

还可以选择的内部功能部件有：HDLC、SDLC、CD-ROM Decoder、Ethernet MAC、VGA controller、DC-DC。可以选择的内置接口有：IIC、SPDIF、CAN、SPI、PCI、PCMCIA。

19、封装问题

ARM 芯片现在主要的封装有 QFP、TQFP、PQFP、LQFP、BGA、LBGA 等形式，BGA 封装具有芯片面积小的特点，可以减少 PCB 板的面积，但是专用的焊接设备，无法手工焊接。BGA 封装的 ARM 芯片无法用双面板完成 PCB 布线，需要多层 PCB 板布线。

20、多核结构 ARM 芯片的选择

为了增强处理多任务与多媒体能力，某些 ARM 芯片中包括两个或多个 ARM

芯核；为了增强数学运算及处理多媒体功能，某些芯片做成 ARM 芯核+DSP 芯核结构；为了提高系统硬件的在线升级能力，某些 ARM 芯片内部集成了 FPGA。

2.2.1.3 ARM 相关技术介绍

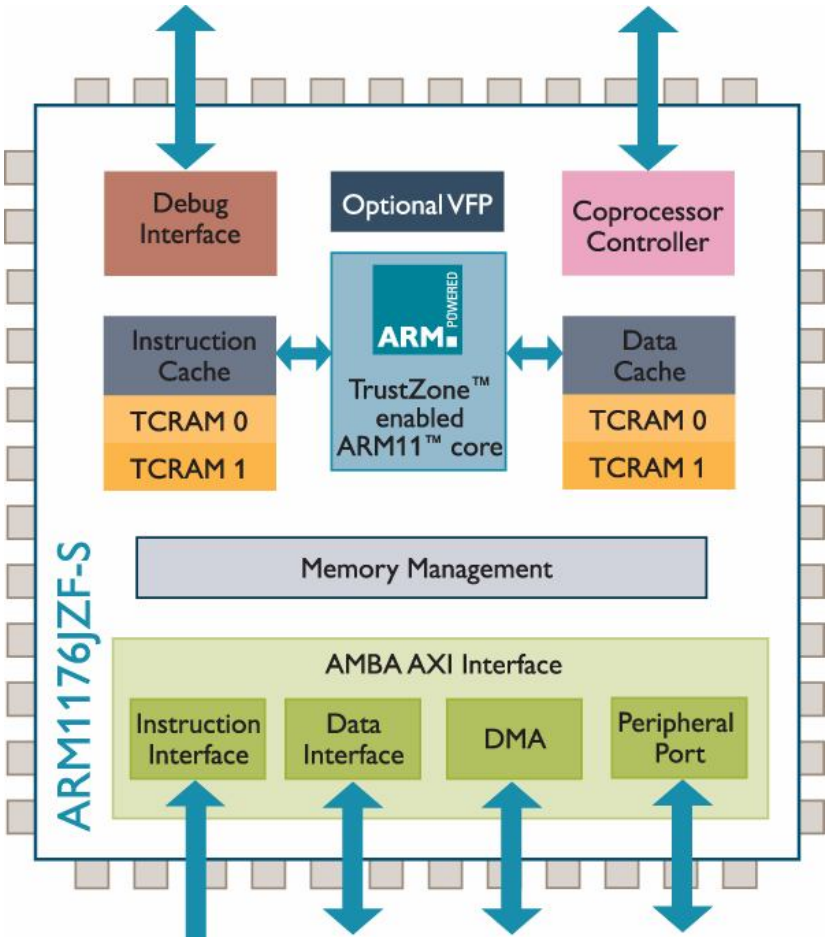

图 2-2 ARM1176JZF-S 的技术架构

Fig.2-2 Technical architecture of the ARM1176JZF-S

如图 2-2：ARM1176JZF-S 的架构是新的 ARM 技术架构，它支持 VFP、TrustZone 等技术，下面介绍 ARM 发展过程中出现的一些技术：

1、Thumb 是在 ARMv4T 之后出现的 ARM 处理器的 16-bit 指令模式。

2、Jazelle 一种用于在处理器指令层次对 JAVA 的编译进行加速的技术。

3、Thumb-2 技术首见于 ARM1156 核心。它扩充了受限的 16-bit 指令集 Thumb，以额外的 32-bit 指令让指令集的使用更广泛。

4、ThumbEE（Thumb Execution Environment）业界称为 Jazelle RCT 技术。它将 Thumb-2 进行某些扩充，使得指令集能特别适用于在所处的执行环境下运行时的编码产生。

5、进阶 SIMD 延伸集有针对多媒体和讯号处理程式进行标准化加速的能力。在业界称为 NEON 技术，是一个结合 64 和 128bit 的 SIMD（Single Instruction Multiple Data 单指令多重数据）指令集。

6、VFP(Vector Floating-point Coprocessor for ARM)向量浮点运算是在协同处理器针对 ARM 架构的衍生技术，它除了提供浮点数基本运算提供支持之外，最有特点的是它的向量（vectors）功能。它能够同时支持 8 组单精度或 4 组双精度浮点数的运算。

7、TrustZone 提供了一种低成本的方案，它为高性能计算平台上的大量应用提供的系统范围的安全方法，其硬件架构旨在提供针对应用的安全框架，从而使设备本身能够抵御可能遇到的众多特定威胁。架构的主要安全目标是支持构建可编程环境，从而对资产的机密性和完整性受到特定攻击进行预防。

2.2.2 温度传感器

DS18B20 是 DALLAS 公司生产的一线式数字温度传感器，它的接线十分方便，封装方式灵活多样，有管道式、螺纹式、磁铁吸附式、不锈钢封装式等多种。封装成后可应用于多种场合，根据用户应用场合的不同可以使用不同的封装方式。这种温度传感器适用于各种狭小空间设备数字测温和控制领域。它小体积封装，封装形式多样，使用方便，耐磨耐碰。

2.2.2.1 技术性能描述

1.独特的单线接口方式，DS18B20 在与控制器连接时只要一条数据通讯引脚便能实现控制器与 DS18B20 的双向通讯。

2.多点接入组网功能简化了温度传感器的多点测温，可以在一根单线上并联多个温度传感器。

3.无需其他辅助元件就可以直接使用。

4.电源范围是 3.0V 至 5.5V，可以仅由数据线供电，而不用独立的电源，供电方式灵活。

5.温度的测量范围为-55℃到 125℃，温度在-10℃到 85℃范围内，测量精度为±0.5℃。

6.温度计的测量精度是可以设定的。用户可通过编程方式将测量精度设定为 9 至 12 位。

7.对于最高的 12 位的温度信号，由模拟信号转换成数字信号最长需要 750 ms。

8.用户可自定义温度报警值，掉电后仍然能永久保存。

9.报警查询命令可以查询到超过编程设置的温度报警条件的传感器设备的标识和地址。

2.2.2.2 DS18B20 内部结构

DS18B20 主要由四部分组成：64 位 ROM、温度传感器、非易失的温度报警除法器 TH 和 TL、配置寄存器。ROM 中的 64 位序列号是出厂前被设置好的，它可以看做是 DS18B20 的地址序列码，每个 DS18B20 的 64 位序列号均不相同。ROM 的作用是使每一个 DS18B20 都各不相同，这样就可以实现一根总线上挂接多个 DS18B20 的目的。如图 2-3 所示，DS18B20 结构。

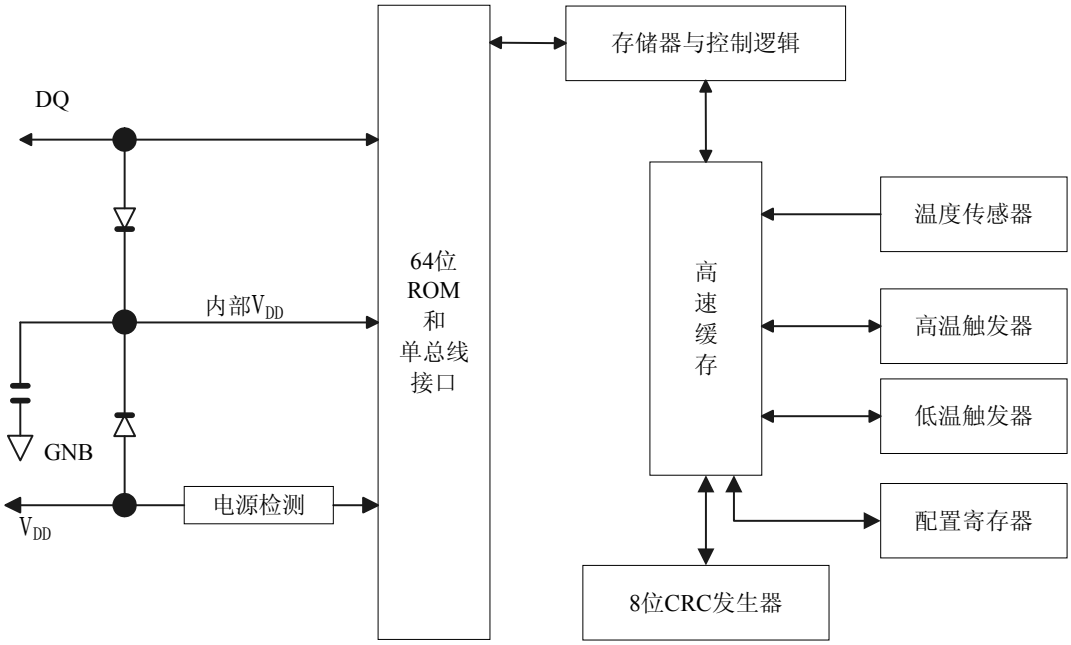

图 2-3 DS18B20 结构

Fig.2-3 DS18B20 structure

DS18B20 管脚排列。如图 2-4 所示。

DS18B20 引脚功能：

GND：接地；DQ：数据输入输出总线；VDD：电源电压；NC：空引脚。

DS18B20 寄生电源工作方式从数据输入输出总线口上获得电源供电。如图 2-5 所示。如果采用独立电源供电，只要在 DS18B20 的 VDD 引脚接 3V 到 5.5V

电源供电即可。

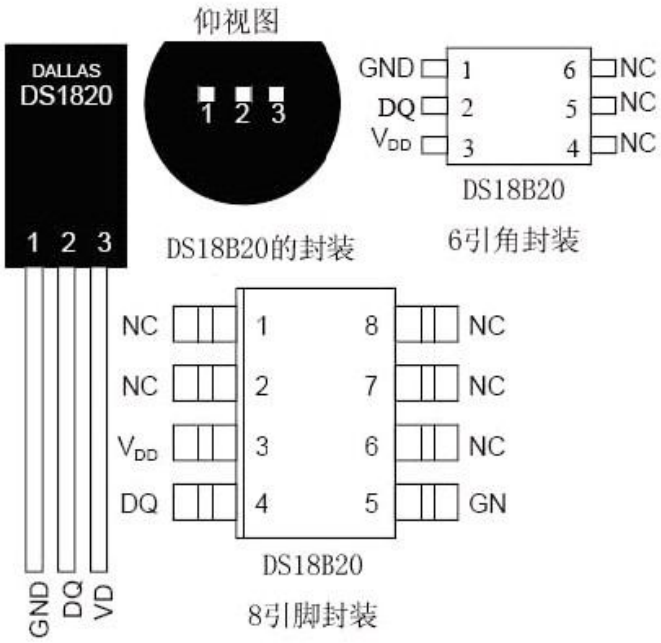

图 2-4 DS18B20 管脚排列

Fig.2-4 DS18B20 pin arrangement

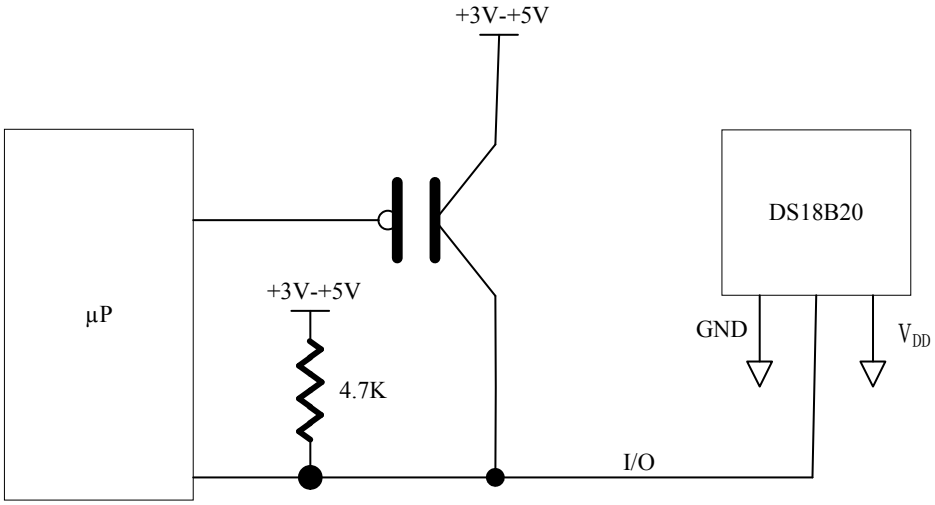

图 2-5 电源工作方式

Fig.2-5 Power work

注意：当温度高于 100℃时，不能使用寄生电源供电，因为此时器件中较大的漏电流会使总线检测到高低电平的可靠性降低，进而导致数据传输时误码率的变大。

2.2.2.3 DS18B20 工作流程介绍

DS18B20 的温度检测功能和数字数据输出功能都集成于一个芯片之上来实现，这使得它的抗干扰力更强。它的一个工作周期包括温度检测与数据处理两部分。为了解它的工作流程，首先要了解 DS18B20 的内部的存储器资源。DS18B20 存储器资源有三种形态。它们分别是：

ROM 只读存储器；RAM 数据暂存器；EEPROM 非易失性记忆体。

DS18B20 工作流程：

S1、控制器发送复位信号给 DS18B20；

S2、DS18B20 发送给控制器存在脉冲，等待数据通信信号；

S3、控制器向 DS18B20 发出 ROM 指令；

S4、控制器向 DS18B20 发出存储器操作指令；

S5、DS18B20 执行对数据的读写等操作。

2.3 软件技术分析

2.3.1 SNMP

根据绪论部分可知简单网络管理协议是对物联网网络终端进行管理的最佳选择，在这里选择它来达到对物联网终端的网络管理的目的。

2.3.1.1 SNMP 介绍

SNMP 即简单网络管理协议是目前最常用的网络管理协议，是 1988 年由 IETF（InternetEngineeringTaskForce）的研究小组为了解决 Internet 上的路由器管理问题而提出的网络管理协议。SNMP 的前身是简单网关监控协议（SGMP），简单网关监控协议最初是用来对通信线路进行管理的。随后，人们对 SGMP 进行了很大的修改，改进后的协议就是 SNMP。在 SNMP 中加入了符合 Internet 定义的 SMI 与 MIB 体系结构。SNMP 的目标是保证管理信息在网络中的任意两点间传送，从而使网络管理员能够在网络上的任何节点上检索相关信息，进行修改并寻找故障；进而完成故障诊断、容量规划及报告生成等工作。它可以在 IP、IPX、AppleTalk、OSI 和其他所用到的传输协议上被使用并提供了一种从网络上的设备中收集网络管理信息的方法。由于 SNMP 在网络管理中的效果，它一推出就得

到了广泛的应用和支持，特别是很快得到了数百家厂商的支持，如 IBM、HP、SUN 等大公司和厂商。现在，SNMP 已然成为网络管理领域中事实上的工业标准，大多数网络管理系统和平台都是基于 SNMP 的[21] 。

随着 SNMP 的发展，目前存在 SNMPv1(简单网络管理协议的第一个正式协议版本，在 RFC1157 中定义)、SNMPv2C(基于共同体的 SNMPv2 管理架构，在 RFC1901 中定义)以及 SNMPv3(通过对数据进行鉴别和加密，提供了许多安全特性)等版本。SNMPv1 有两个主要的缺点：一是效率不高，管理功能不够完善；二是缺乏有效的安全机制[22] 。针对这些缺点 IETF 于 1993 年发布了 SNMPv2 通过扩展数据类型、增加协议操作类型等方法增强了管理功能，同时在安全方面也提出了解决方案。SNMPv3 在先前的版本地基础上加入了保证 SNMP 本身安全的因素增加了安全性和远程配置能力。

SNMP 事实上是指一系列网络管理规范的集合，它是由一系列协议和规范组成的，其中包含一些数据结构的定义、协议本身和一些相关的概念[23] 。这些协议和规范提供了从网络设备中获取网管信息的一种方法。SNMP 是适用于互联网络设备的网络管理框架，主要由 3 个部分组成：管理信息结构(Structure of Management Information，SMI)，管理信息库(Management Information Base，MIB)以及 SNMP 协议。为了对网络进行有效管理，SNMP 在模型中定义了大量的变量用来描述网络上硬件和软件的运行状态和统计信息。在 SNMP 中定义的这些变量叫对象，所有对象都在 MIB 中定义。MIB 是被管理对象的结构集合，这些被管对象由设备所维护管理。被管理对象按照树形结构分层组织，每个对象都由一个特殊对象类型的对象唯一标志，这个特殊对象由一串整数构成。管理信息结构 SMI 是用于定义通过网络管理协议可访问的对象的规则，MIB 中使用的数据类型及网络资源的命名与表示也是在 SMI 中定义的[24] 。SNMP 协议则是运行于网管器上的管理进程和运行于被管设备的代理进程之间的通信协议。它的设计原则是尽量简单、尽量少增加网络负担。SNMP 协议利用 UDP 协议的 161/162 端口。SNMP 协议操作有 3 种：一种是管理器与代理之间的请求-响应操作；另一种是管理器和管理器之间的请求-响应操作；第三种是代理对管理器的无确认通行操作，即 Trap 操作。

2.3.1.2 MIB 定义

在系统管理模块中，对网络资源信息的描述极其重要。管理信息库(MIB)作

为网络管理中的被管资源，是以对象的形式表示的，每个对象表示被管资源的某一个方面属性，对象的集合形成了 MIB 模块。每个 MIB 对象或变量记录了网络的状态、通讯流量数据、设备配置信息、发生差错次数以及内部数据结构的当前内容等信息。网络管理者通过对 MIB 库的存取访问，来实现网络管理的五大管理功能[25] 。

MIB 对象的定义使用 ASN.1 语言表示。ASN.1 是一种描述结构化客体的结构和内容的语言，SMI 规定了对象信息的编码采用基于编码规则从而对对象信息统一编码，使数据的描述与系统和厂家无关。这种统一的信息编码使得网络中所有的终端都能清楚的理解网络中所传输的信息。ASN.1 的基本单位是模块，用于定义一个抽象类型 ASN.1 模块，实际上模块是由一组类型定义和值定义组成，模块定义的基本形式用 ASN.1 描述如下

<moduleIdentifier> DEFINITIONS::=BEGIN

EXPORTS

IMPORTS

AssignmentList

END

moduleIdentifier 是模块名，注意：模块名的首字母必须使用大写字母。

EXPORTS 部分是用于定义此模块中那些可以被其它模块所引用的一组类型或定义。

IMPORTS 部分规定了此模块中某些定义是引自其它哪些模块。

AssignmentList 部分是本模块定义的所有的类型、值和宏定义。

MIB 对象定义格式是 SMI 规定的，对象定义就是模块定义中的 AssignmentList 部分，是用来定义类型和值的。用 ASN.1 描述如下：

OBJECTNAME BOJECT-TYPE

DESCRIPTION:(description)

SYNTAX:(syntax)

MAX-ACCESS/ACCESS:(access)

STATUS:(status)

::={(Parent)number}

OBJECTNAME 是对象名，应具有全局唯一性；

BOJECT-TYPE 是节点对象关键字；

SYNTAX 是被管对象类型关键字，(syntax)是对象类型；

MAX-ACCESS/ACCESS 是被管对象的访问方式关键字，MAX-ACCESS 子句与 ACCESS 子句类似，但是 ACCESS 子句是用在 SNMPv1 中的访问方式关键字，MAX-ACCESS 子句是用在 SNMPv2 中的访问方式关键字。(access)是被管对象的访问方式，SNMPv1 中可取值有：read-only、read-write、write-only、no-accessible，SNMPv2 中少了 write-only，多了 read-create、Accessible-for-Notify。

STATUS 是被管对象状态关键字，(status)是被管对象的当前状态，在 SNMPv1 中可取值有：mandatory、optional、obsolete、deprecate；在 SNMPv2 中可取值有 current、obsolete、deprecate。

mandatory 为该对象的状态是必备的；optional 为该对象的状态是可选的；obsolete 为该对象的状态是不再使用；deprecate 使用该值标记的 MIB 的那一部分标记为过时，只有那些被标记为“obsolete”的对象会从现行版本中删除，同时在新的 MIB 版本中标记为“deprecate”；current 为该对象的状态是当前可用的。

DESCRIPTION 是对被管对象进行描述的关键字，主要描述该对象的功能和特征。(description)是其文本描述，在::={(Parent)number}中，Parent 表示位于 MIB 树中的父节点，number 表示子节点。

2.3.1.3 MIB 树

每个 MIB 对象都由对象标识符(OID)来惟一确定和标识。OID(object identifier)是一组由数字组成的，并且数字中间用句点隔开，它指出了该节点在树中的准确位置。每个 MIB 对象都是一个带标号的节点，每个节点可以同时用数字和文本描述。一个带标号的节点可以拥有子树，它的子树也是由带标号的节点组成，如果一个节点没有子树，它便是叶子节点，也称为对象。每个对象都是由一个标号序列惟一确定的，这个标号序列是从树根到该对象的节点上的路径的标号序列[25]。整个 MIB 树构成一个层次树，结构如图 2-6 所示。

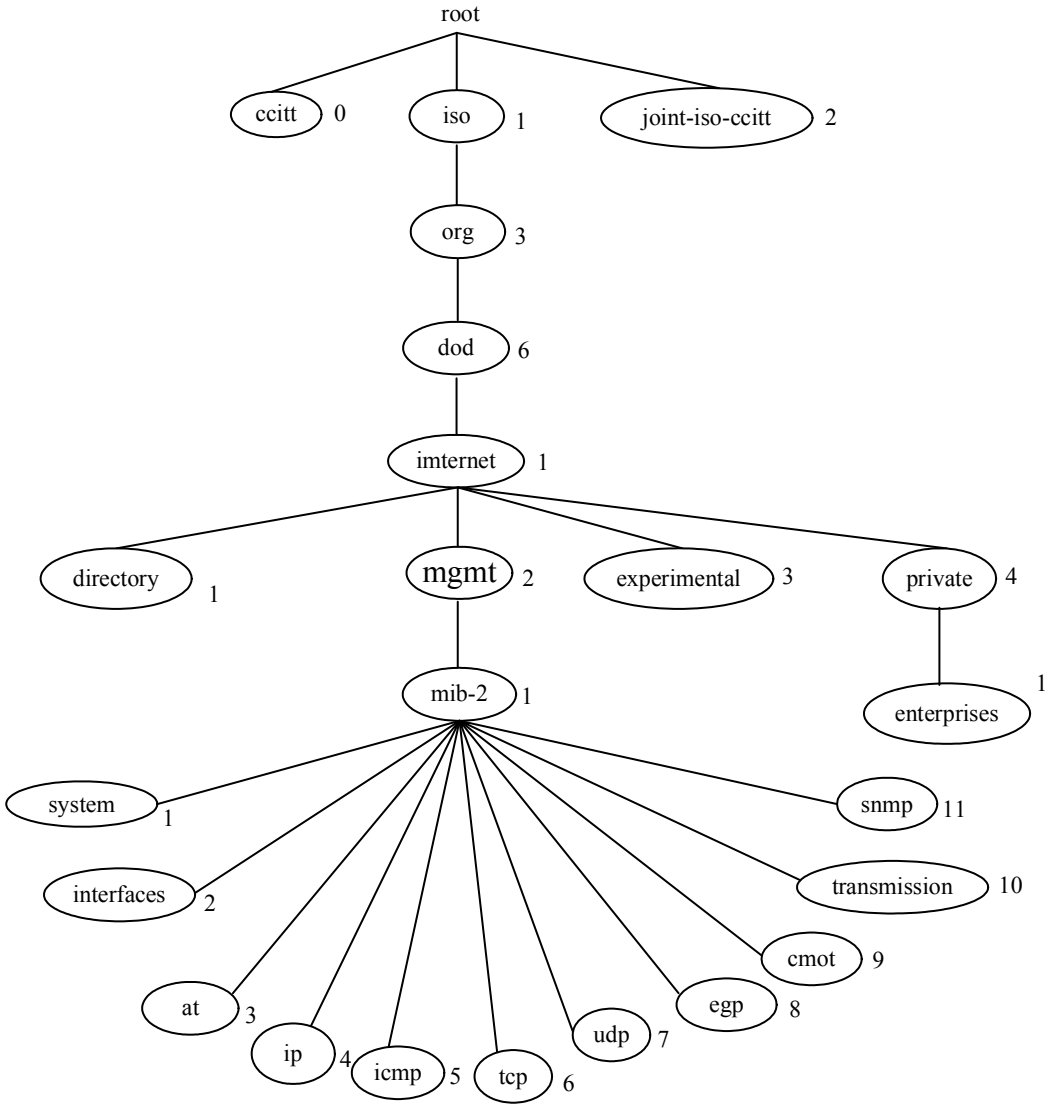

图 2-6 MIB 树型结构

Fig.2-6 MIB tree structure

2.3.1.4 MIBs 实现要求。

对于双协议栈，需要支持 1998 年定义的 IPv6 与 IPv4 分离管理的 MIB 库（RFC2452/2454/2465）或以后定义的统一管理 MIB 库（RFC2851、RFC3291）。对于 SNMP，出于对未来网络管理时对 SNMPv3 需要的考虑以及对已有投资的保护需要支持 SNMPv3、SNMPv2、SNMPv1 三个版本的相关 MIB。

2.3.1.5 SNMP 协议操作

SNMP 规定了 5 种 PDU 用来在管理进程和代理进程之间交互，PDU 就是指

SNMP 报文。SNMP 的五种网管的操作原语为：

Get-request 操作：从代理进程中获得参数的值。

Get-next-request 操作：从代理进程中获得紧跟当前参数值的下一个参数值。这个操作主要用来遍历网络设备 MIB 库的某个对象的一系列参数。

Set-request 操作：设置代理进程的参数值。

Get-reponse 操作：返回的参数值。这个操作是由代理进程发出的对是上面三种管理器进程操作的响应操作。

Trap 操作：异常情况的发生由代理进程主动发出向管理进程报告的报文。

前面的 3 种操作是由管理器发给代理的，后面 2 种操作则是代理发给管理器的。在 SNMP 操作时，代理一般是用端口 161 接受 Get-request、Get-next-request 或者 Set-request 报文，而在管理器一般是用端口 162 来接收代理的 Trap 操作的信息[25] 。

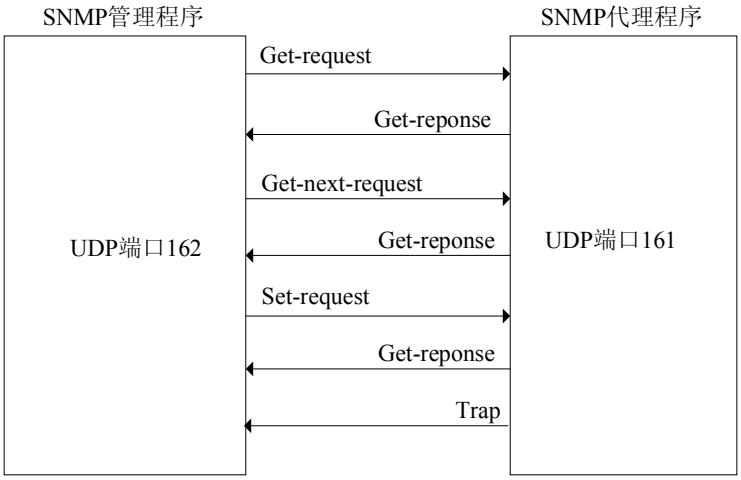

图 2-7 SNMP 报文操作

Fig.2-7 SNMP packet operation

2.3.2 嵌入式操作系统选择及内核裁剪

2.3.2.1 嵌入式操作系统选择

嵌入式操作系统 EOS（embedded operating system）负责嵌入式系统的全部软、硬件资源的分配、调度、控制、协调并发活动，它必须体现其所在系统的特征，能够通过装卸某些模块来达到系统所要求的功能[26] 。嵌入式操作系统伴随着嵌入式系统的发展经历了 4 个阶段：第一阶段是无操作系统的嵌入算法阶段；第二阶段是以嵌入式 CPU 为基础、以简单操作系统为核心的嵌入式系统阶段；

第三阶段是通用的嵌入式实时操作系统阶段，是以嵌入式操作系统为核心的嵌入式系统，这是目前广泛应用的操作系统；第四阶段是基于 Internet 嵌入式系统阶段，这是一个正在迅速发展的阶段。

下面介绍几个嵌入式操作系统，根据 2.1.1 节对 ARM 的介绍和选择，将从中选择一款操作系统作为开发本系统的目标平台：

1. uC/OS—II

uC/OS—II 是赫赫有名的开源嵌入式 OS（Operating System），但如果用于商业目的，需要授权。uC/OS—II 内核简单清晰，是极好的学习嵌入式实时操作系统的入门材料。近来增加了 uC/GUI 图形界面，uC/FS 文件系统，uC/TCP 网络功能，这些都是要收费的。进行简单的开发还是不错的选择。在 8 位的 51 系列单片机用的很多，16 位、32 位也支持。

2. FreeDOS

FreeDOS，是开源的兼容 DOS。自从微软抛弃 DOS，专门致力于 Windows 操作系统的开发之后，开源社区就开发了 FreeDOS。DOS 的几个缺点：单进程，网络功能弱。用于 DOS 免费的网络协议栈已经有了。

3. 嵌入式 Linux

Linux 如此有名，但因为实时性问题，只能用在对实时性要求不高的嵌入式系统中[27] 。不过有几家大公司比如风河，在开发高实时性的 Linux。在服务器和嵌入式方面 Linux 的前途越来越好，但是对于桌面来说，实在难说。Linux 是 32 位的，在 ARM9 上使用的比较多。

4. uCLinux

uCLinux，是开放源代码的嵌入式 Linux 的榜样。它主要是针对没有存储管理单元 MMU 的处理器而设计的，简单的说就是主要用于没有虚拟内存的处理器[28] 。虽然它的体积很小，却继承了 Linux 的大多数的优点：稳定、良好的可移植性、优秀的网络功能、对各种文件系统完备的支持和标准丰富的 API。32 位 ARM7TDMI 的首选。

5. RTLinux

RTLinux，Linux 的实时扩展，针对实时的需求 RTLinux 提出了精巧的内核，把标准的 Linux 核心作为实时核心的一个进程，把实时 API 作为驱动模块加载到内核，两者一起调度，通过实时 FIFO 进程间通讯。

6.红旗嵌入式 Linux

由北京中科院红旗软件公司推出的嵌入式 Linux，由于有中科院计算所的强大科研力量做后盾，它也是目前我国做得较好的一款嵌入式操作系统。

嵌入式操作系统还有很多种，如 VxWorks、Nucleus、Windows CE、QNX 等。在这里就不再做更多的介绍。

Linux 的内核从 2.4 版就已经支持了 IPv6，同时，Linux 作为开源的操作系统，也便于向不同的硬件平台移植。本课题选取的 ARM11 硬件开发平台需要针对它采取更加灵活的方式完成设计与实现，从程序移植的难易程度到各种可参考的资料容易获得与否等。考虑到种种原因，嵌入式 Linux 作为本课题最终选择的嵌入式开发操作系统平台。由于现有的网络基础设施运行的大都是 IPv4 协议栈，IPv6 作为下一代互联网协议，是未来网络发展的趋势，物联网也需要对 IPv6 的支持，才能够大规模的投入使用[29] 。IPv4 向 IPv6 过度的主要技术是双协议栈技术。本课题通过内核裁剪来支持双协议栈技术。

2.3.2.2 嵌入式 linux 操作系统内核裁剪

物联网设备的硬件和软件要更高效率地设计，量体裁衣、去除冗余，力争在小型设备上实现更高的性能，才能更具有竞争力。由于嵌入式资源极其宝贵，在进行嵌入式开发时，首先要根据自己的应用需求和硬件结构等硬件信息裁剪内核，将不必要的代码尽量裁剪掉，以免浪费空间和资源，然后再编译产生对应目标板的 linux 嵌入式内核。通过对 Linux 内核源码及组成内核的分析，得到内核模块中可以裁剪的部分：

①系统多余的进程。Linux 内核中的一些模块对于嵌入式系统并不适用，可以剔除内核对这些进程的创建和调度机制。

②平台无关代码和多余硬件相关代码。根据硬件平台裁剪掉内核 arch 目录下与该平台无关的处理器支持代码; 另外，Linux 内核支持很多硬件及其周边设备，并不是所有嵌入式系统都需要，可以根据实际情况删除那些无用的硬件驱动程序的相关代码。

③异常处理函数。Linux 内核提供的一些异常处理函数是嵌入式系统所不需要的，可以根据具体情况删除不需要的函数代码。

④内存管理。Linux 的内存管理采用的是基于分页式的虚拟内存管理机制，在嵌入式系统中并不适用可以剔除，直接使用实际内存。

⑤文件系统。Linux 内核支持多种文件系统，可以裁剪掉不需要的文件系统

来节省存储空间。

⑥网络协议栈。嵌入式系统可根据需求剔除系统中无需支持的网络协议栈，仅保留需要的网络协议栈。

裁剪过程简单说明如下：

首先，下载 linux 源代码，并为它打上对应的补丁，使它能够成为支持各种 CPU 的内核。然后，进入 linux 源代码文件夹，可以使用 make menuconfig 或者 make config 等配置内核命令进行 linux 内核的配置。最后，编译内核产生目标板对应的内核程序和二进制文件。下面是使用 make menuconfig 命令配置 linux-2.6.36.2 内核时的选项。并对一些选项做了简单说明。

配置硬件过程需要了解其中的内涵，这个过程复杂而枯燥，可以直接使用由嵌入式开发板供应商提供相应的配置好的嵌入式 linux 内核，然后再根据自己的应用需求配置所需要的东西，比如在需要配置 IPV6 和 IPV4 的支持时，这就要查阅相关资料，通过查阅资料我知道在[*] Networking support ---->中配置相应的选项就可以做到这些。更具体的配置内容需要查阅相关文档。下面仅仅对内核配置的一些模块做简单的说明。

General setup ---> 常规安装选项

[*] Enable loadable module support ---->可引导模块支持建议作为模块加入内核

-*- Enable the block layer ---->块设备支持，使用硬盘/USB/SCSI 设备者必选

System Type ---->系统类型

Bus support --->总线支持

Kernel Features --->

Boot options --->

CPU Power Management ---> CPU 电源管理选项

Floating point emulation --->

Userspace binary formats ---->支持的可执行文件格式

Power management options ---->电源管理选项

[*] Networking support ---->网络支持

Device Drivers ---->设备驱动程序

File systems ---->文件系统

Kernel hacking ---->内核 hack 选项，普通用户是用不着这个功能的

Security options ---->安全选项

-*- Cryptographic API ---->提供核心的加密 API 支持.这里的加密算法被广泛的应用于驱动程序通信协议等机制中.子选项可以全不选，内核中若有其他部分依赖它，会自动选上

Library routines ---->库子程序

Load an Alternate Configuration File 读入一个外部配置文件

Save an Alternate Configuration File 将配置保存到一个外部文件

2.4 小节

本章开始确定了物联网终端模型，通过对物联网终端相关硬件及软件技术的分析与研究，选择了 ARM、温度传感器作为本课题研究的物联网终端的硬件要件，选择嵌入式 linux 作为嵌入式操作系统平台，用作开发物联网终端相关应用平台，同时选择 SNMP 管理模型为物联网终端管理模型，NET-SNMP 软件开发包作为开发 SNMP 的开发平台，为物联网终端系统架构的设计打下基础。

3 物联网终端网络管理设计与实现

根据第 2 章建立的物联网终端模型和对硬件、软件两个方面技术的分析与研究，笔者已经找到了实现物联网终端网络的硬件架构和软件结构。下面将具体介绍物联网终端系统组成、硬件结构、软件架构、数据处理流程及管理实现的主要工作等。

3.1 物联网终端系统结构设计

3.1.1 系统组成设计及实现概述

根据上一章的介绍，将案例终端如此设计：它包括硬件（ARM11 核心板）、系统软件（嵌入式操作系统）、数据采集设备（温度传感器）等几个部分组成，下面的工作就围绕这个终端来进行。

首先，要考虑的是资源，要实现这个模块，有如下资源可以利用：

硬件方面：forlinx 公司开发的 ok6410 开发板、DS18B20 温度传感器；

软件方面：系统软件选择的是嵌入式 linux 操作系统和用于开发 SNMP 代理的 NET-SNMP 软件包；

资料方面：forlinx 公司提供的大量的 ok6410 开发板相关资料及开发技术支持与设计文档、网络上大量关于 SNMP 开发的参考资料、NET-SNMP 网络主页的大量可参考资料、以及针对其他 ARM 平台编写的 DS18B20 温度传感器驱动和它的硬件资料、工作原理等。

在开发之前，要先设计物联网终端模块具体的硬件结构和软件架构。然后，才可以着手网络模块开发的工作，这里主要做如下工作：为了做开发建立一个开发平台，其中要能够编译 linux 内核、建立完整的交叉编译环境、安装好 NET-SNMP 软件开发包。具体的开发工作如下：编译内核为目标板生成二进制编码内核文件、为目标板编译 uboot 和文件系统、烧写系统到目标板、为传感器编写驱动程序、开发 SNMP 代理、移植代理程序到目标板、测试。

3.1.2 物联网终端硬件架构设计

根据上面的对 ARM 的介绍和温度传感器的介绍，综合考虑各种技术方面的因素，为物联网网络模块单元和其支持的应用留有可以多方面扩展和支持的余

地，使其成为具有开放性的设计，另外考虑到本设计的初衷是为了物联网传感器接入网络并进行网络管理提供解决方案，由于 DS18B20 温度传感器具有较好的代表性，所以选择以 DS18B20 温度传感器作为案例，为以后其他传感器的设计提供有价值的参考。

如图 3-1 所示：物联网终端硬件主要由控制器（ARM11）和智能数据采集设备组成，其中控制器（ARM11）通过有线网络或者无线网路连接到 internet 上，温度传感器可以直接焊接到控制器（ARM11）硬件开发板上。选择的 ARM11 的开发板是飞凌公司开发的 ok6410 嵌入式开发板，其核心是 ARM1176JZF-S，支持 wifi 和有线网络。

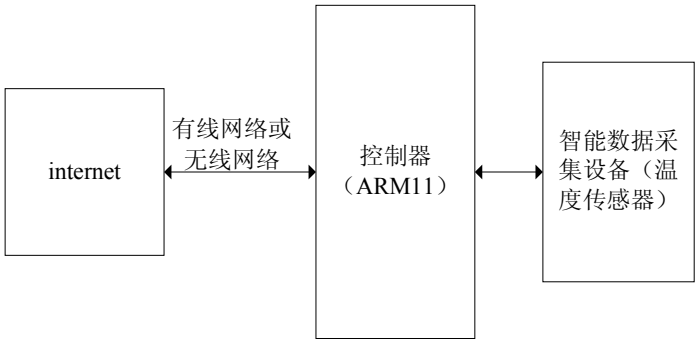

图 3-1 物联网终端硬件结构

Fig.3-1 Internet of Things terminal hardware structure

3.1.3 物联网终端软件结构设计

根据上面内容的介绍，本系统所要设计的软件结构如图 3-2 所示。

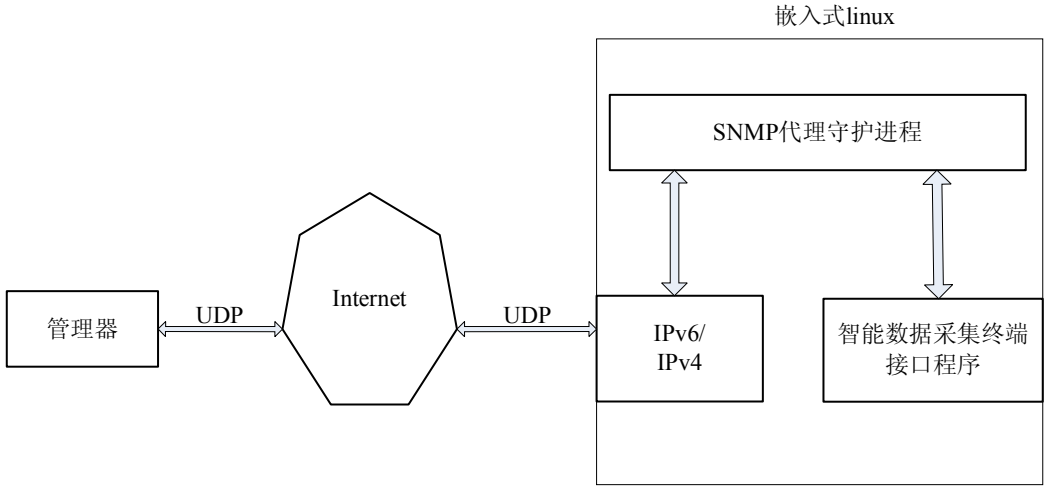

图 3-2 软件结构

Fig.3-2 Software architecture

为了更容易支持 IPv6/IPv4 双协议栈结构，选择嵌入式开发的操作系统是嵌入式 linux，当然这只是选择它的其中一个原因。由于需要更好的对物联网的传感器节点进行有效的网络管理，对简单网络管理协议（SNMP）的支持是必须要考虑的一个选择，在开发相关 SNMP 代理时需要使用功能强大的 NET-SNMP 作为开发平台，同时在 ARM 和传感器相互通信时需要编写 ARM 下的温度传感器驱动。管理器和代理之间采用 UDP 协议通信。

3.1.4 物联网终端数据处理流程

物联网终端数据处理流程如图 3-3 所示。

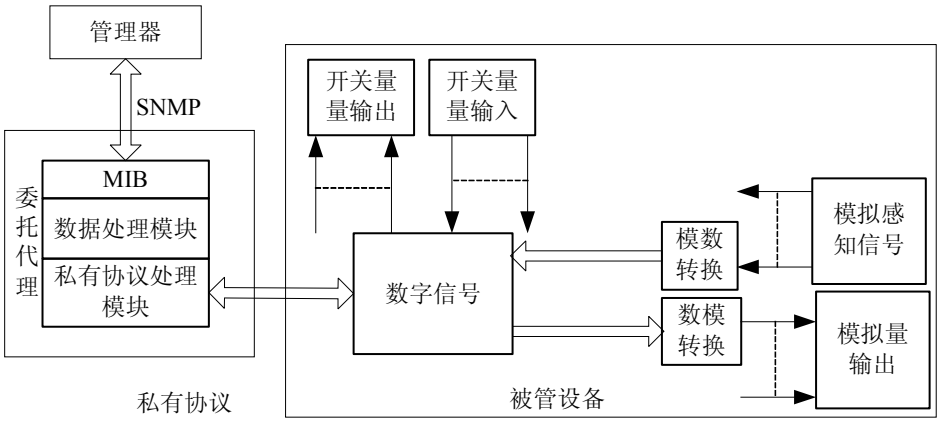

图 3-3 物联网终端数据处理流程图

Fig.3-3 Terminal of Internet of things data processing flow chart

智能数据采集设备（即图中的被管设备）是本课题中作为物联网感知层的设备，它主要用来产生数字信号，获取自然界中数据信息。但是有很多智能数据采集设备不能够直接产生数字信号，而产生的是模拟信号，为了能够处理这些数据，需要将模拟信号转换成数字信号，在数据处理后，为了使对该设备返回来的数据能够起作用，也需要将数字信号转换成模拟信号，本课题中的被管设备（DS18B20 温度传感器）在其内部有数模转换和模数转换的器件，能够直接发出数字信号给代理。

在本课题的数据处理流程中，核心是委托代理。之所以选择委托代理的方式进行，是因为被管理设备是非 SNMP 设备，这时就需要以委托代理（Proxy Agent）作中介，完成管理站和被管设备之间的协议转换功能。委托代理当接收到来自管理站的查询和设置命令时，委托代理把所操作的 MIB 对象映射为被管设备的相

应管理信息，将 SNMP 命令翻译成与被管设备通信的私有协议。所以，委托代理在保护用户已有投资的基础上，拓宽了用户对自身设备的管理能力。同时，委托代理能够精简来自被管设备的私有协议数据，降低网络流量，也使得管理站能更有效地处理返回信息。

委托代理主要包括 MIB、数据处理模块、私有协议处理模块。私有协议处理模块是用来处理与非 SNMP 的被管设备进行通信的，由于委托代理支持多种被管设备，而不同的设备所支持的协议是不同的，所以私有协议处理模块要根据需要支持的设备实现对应的私有协议，在本文中私有协议处理模块表现为设备的驱动程序。

数据处理模块提供每个 MIB 变量的处理函数，是协议转换的具体实现并对 MIB 库数据进行更新。根据 PDU 类型和变量的 OID，构造该变量相对应的被管设备私有协议中的查询或设置命令，通过私有协议处理模块发送到被管设备。如果是查询命令，则从应答报文中抽象出 MIB 变量的值后更新 MIB。如果是设置命令，在收到应答报文后，判断是否设置成功，若成功则修改 MIB 中对应变量的值。最后发送 Response 报文到管理站。由于一条私有协议命令中包含的管理信息可能被分为多个 MIB 变量，那么对其中一个变量进行操作时，就同时也要对其他变量进行操作。如果是查询命令，那么接收到来自被管设备的应答后，对所有 MIB 变量的值进行更新。如果是设置命令，则将管理站设置所操作变量的值和其他变量在 MIB 中的值构成一条设置命令发送到被管设备。

管理站可以根据网络需求设置委托代理的查询操作模式（快速模式和实时模式），指示进行查询操作时委托代理返回变量值的操作方式。快速模式是委托代理将周期性轮询被管设备时存储在 MIB 中的值直接返回给管理站，而不需要访问被管设备，减少被管设备的处理开销和通信时间。实时模式是委托代理将访问被管设备的实时信息返回给管理站，管理站需要花费时间等待委托代理和被管设备之间的通信。通过私有协议处理模块发送每个 MIB 变量对应的查询命令，收到应答报文后更新 MIB，并判断是否发送 Trap 报文。同时，从应答报文中检查被管设备的配置变化情况，以判断是否需要对 MIB 进行动态增添和删除。

3.2 构建嵌入式 linux 开发环境

本设计的开发环境由宿主机操作系统环境、交叉编译环境、NET-SNMP 开发平台、目标机嵌入式 linux 操作系统几个部分组成。现在开始构建开发环境。

3.2.1 搭建开发环境、建立交叉编译环境

3.2.1.1 搭建宿主机操作系统、建立系统环境

在开发之前，搭建开发环境是必不可少的一个步骤，适当的开发环境会为开发带来很多便利。开发环境并没有一个固定的样式，可以根据自己的需要搭建相应的开发环境。选择 Ubuntu 作为开发环境，主要是因为它是一个以桌面应用为主的 Linux 操作系统，它相对于其他版本的 Linux，拥有很多优点。首先，安装系统非常简单，只需要非常少的设置即可，完全可以与 Windows 这样的桌面系统的安装过程想媲美；其次，它拥有很人性化的图形界面，还模仿了在 xp 下常用的快捷键，惯用了 windows 的人在这方面不会感到太陌生；还有，在安装和升级程序时，往往会因为其他 linux 系统程序之间的依赖关系而大伤脑筋，然而对于 Ubuntu Linux 来说可以通过网络自行安装程序和它依赖的文件包，从此不必再为了这个问题而苦恼。综合考虑上面诸多因素，选用 Ubuntu Linux。

从 Ubuntu 官网：http://www.ubuntu.org.cn 上可下载各个版本的 Ubuntu。当然，官网上也能找到各种关于 Ubuntu 的相关信息。另外 Ubuntu 官方论坛：http://forum.ubuntu.org.cn/也可以找到大量的实用的 Ubuntu 资源及中文论坛。Ubuntu 也有官方的英文论坛。这些资料都对的系统搭建有帮助。

首先下载的操作系统版本是 Ubuntu9.10，在安装时可以直接安装到电脑上，也可以安装在 windows 操作系统的虚拟机上，不过在进行嵌入式开发时，最好能够使用直接安装的 Ubuntu，虚拟的操作系统对于嵌入式开发有一定的负面影响。如果对 windows xp 系统比较熟悉，初步接触 Ubuntu，最好还是选择使用虚拟的操作系统。

选择安装系统的虚拟机是 VMware，具体的安装虚拟机过程就不做介绍了，仅对 Ubuntu9.10 的安装过程做简要说明：

S1、打开 VMware 创建一个虚拟机；

S2、将从官方网站上下载的 Ubuntu9.10 安装文件载入 VMware 工作站中，开始安装 Ubuntu9.10 到虚拟机中；

S3、开始安装后，系统提示选择安装使用的语言，这里直接选择中文为安装语言；

S4、接着选择安装 Ubuntu；

S5、选择操作系统语言，点击前进。这里可根据自己的需求选择语言，Ubuntu

支持多种语言，也可以在安装 Ubuntu 完成后更新语言包；

S6、系统自动同步操作系统时间；

S7、选择所在地；

S8、选择键盘布局；

S9、硬盘空间以及挂载点的分配。这里选择默认；

S10、填写用户名及用户密码，将启动方式选择为“自动登录”；

S11、最后是显示一些安装的配置信息。点击“安装”；

S12、安装完成后，点击“现在重启”。

系统安装完后要设置好网络，由于在开发过程中需要 root 用户权限的地方很多，所以直接设置 root 用户自动登录，这些设置在此略过，不在做具体介绍。

3.2.1.2 交叉编译环境的建立

在裁减和定制 Linux 用于嵌入式系统开发之前，由于嵌入式系统一般存储量大小有限，通常需要要在自己的 pc 机上建立一个用于编译运行于目标机程序的交叉编译环境[30] 。它是一个综合开发环境，由编译器、连接器和解释器组成。binutils、gcc 及 glibc 等几个部分组成交叉编译工具。建立一个交叉编译工具链要经历一个很复杂的过程，如果你不想亲自体验这个复杂的编译过程，在网上有一些编译好的交叉编译工具链可以直接下载使用。由于本文的目的不是做这个工作，现仅将过程简单说明，交叉编译环境的建立起来比较复杂，且费时费力，也没有多少工作上的意义。所以本项目的交叉编译环境是通过网络下载到的现成的交叉编译环境，这里仅简单介绍一下这个过程：

S1、下载源文件、打补丁并建立编译的目录

选定各个软件版本号

建立相应的工作目录

输出在编译过程中常用到的环境变量或者直接有绝对路径省去这一步

建立对应的编译目录

S2、建立内核头文件

解开 linux 内核源代码

为 Linux 内核打上对嵌入式开发板支持的补丁

编译内核生成头文件

S3、建立二进制工具（binutils）

S4、建立初始编译器（bootstrap gcc）

接着就是配置 boostrap gcc

接着编译并安装 boot-gcc

S5、建立 c 库(glibc)

然后进入源文件 build-glibc，目录配置 glibc

编译和安装 glibc

S6、建立全套编译器（full gcc）

编译和安装 full gcc

3.2.2 NET-SNMP 及其安装配置

NET-SNMP 是一套开放源代码的简单网络管理协议（Simple Network Management Protocol）软件包。它可以同时使用 IPV4 和 IPV6 并同时支持 SNMPv1，SNMP v2c 与 SNMP v3 三个版本。

3.2.2.1 NET-SNMP 软件结构

NET-SNMP 软件包结构包括：

①命令行应用程序：

从支持 SNMP 的设备上获取信息，可以使用单一的请求（snmpget 命令，snmpgetnext），也可以使用多个请求（snmpwalk 的，snmptable，snmpdelta）。

使用 snmpset 命令为支持 SNMP 功能的设备进行配置。

从支持 SNMP 的设备上获取固定的信息（snmpdf，snmpnetstat，snmpstatus）。

实现 MIB OID 的数字和文字形式之间的相互转换，并显示 MIB 的内容和结构（snmptranslate）。

②一个基于 Tk/perl 的图形化的 MIB 浏览器（tkmib）。

③一个用于接收 SNMP 消息的应用程序（snmptrapd）。这些消息可以选择记录在系统日志，NT 事件日志，或纯文本文件等日志中，或者转发给另一个 SNMP 管理系统，或传送给一个外部应用程序。

④一个用于响应 SNMP 信息管理程序（snmpd）查询操作的可扩展代理。它包括对广泛的 MIB 信息模块的支持，通过动态加载模块、外部脚本和命令来扩展代理，同时支持 SNMP 多路复用（SMUX）和代理扩展协议（AgentX）。

⑤基于 C 和 perl 的 API 函数库，它可用于开发新的 SNMP 应用。

Net-SNMP 可以用于许多 Unix 和类 Unix 操作系统，甚至支持 Microsoft Windows。但是它所能使用的功能具体取决于操作系统的类型。

3.2.2.2 NET-SNMP 安装

NET-SNMP 包目前可以移植的版本包括各种 UNIX、LINUX、Windows 等不同版本。

目前 NET-SNMP 包的安装主要有两种方式：源程序编译安装、二进制文件安装，源程序安装可以直接从 NET-SNMP 的官网下载源码、然后按照 configure、make、make install 三个步骤即可编译安装[31] 。二进制安装可以到官网及各操作系统厂商的网站下载。

对于 Ubuntu 来说可以使用如下命令直接安装 NET-SNMP：

sudo apt-get install snmpd

然而由于以后要用到 NET-SNMP 的源码进行交叉编译和移植，所以还是要下载源码，现在最新的版本是 5.7.1，下载源代码 net-snmp-5.7.1.tar.gz。其遵循的安装过程为：

gzip -d net-snmp-5.7.1.tar.gz（解压缩）

tar -zxvf net-snmp-5.7.1.tar（解包）

然后进入源码目录使用 configure 配置 SNMP，配置的详细内容这里不在写，在后面的代理程序移植过程中会有更具体的介绍，最后使用 make、make install 命令编译并安装程序。安装完后可以运行客户端程序并不能直接运行代理，还要配置好 SNMP 代理并运行代理。

3.2.2.3 NET-SNMP 配置

①配置 snmpd.conf

既可以使用/usr/local/bin/snmpconf 对它进行配置，也可以直接修改 snmpd.conf。

如果使用/usr/local/bin/snmpconf 对它进行配置，需要确认 perl 的安装路径，并更改 snmpconf 的第 1 行 PERL 脚本 BIN 路径，因为它是由 perl 编写的，需由 perl 编译运行。snmpconf 脚本主要是用来配置 snmp.conf、snmpd.conf、snmptrapd.conf 三个配置文档的。运行 snmpconf 命令后是一个可以交互的设置过

程，对于 snmpd.conf 主要设置如下内容：

System information setup——包括代理的位置、联系人信息等；

Access control setup——访问控制设置，主要配置团体名、团体组和相应的权限等；

Agent Operating Mode——可以访问本地代理的 IP 及本地代理的端口号等。如图 3-4 所示，配置 snmpd.conf 的交互界面选项。

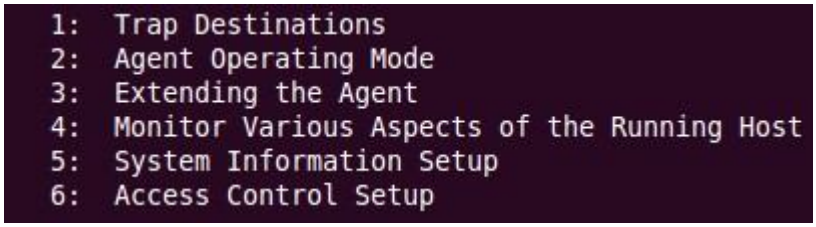

图 3-4 配置 snmpd.conf 的交互界面选项

Fig.3-4 Interface options of configur snmpd.conf

如果直接修改 snmpd.conf 文件中的参数信息对 snmpd.conf 进行配置。做法如下：

将安装包中的 EXAMPLE.conf 复制到/etc/snmp 文件夹下，并重命名为 snmpd.conf。使用以下命令复制：

cp EXAMPLE.conf /etc/snmp/snmpd.conf

定义安全体

在 snmpd.conf 文档中找到如下字段，修改为需要的参数。

sec.name source community

com2sec local localhost public

参数简介：

sec.name：安全体名称

source：定义请求的来源。在 net-snmp 中用来对来源 IP 加以控制，但这个特性不是 SNMP 规定的，是 net-snmp 扩展的。

community：共同体名称

配置完这些信息，代理程序就可以使用了，其它参数可以使用默认值。更具体的配置下面做一个简单介绍。

其它 Access Control（访问控制）配置介绍如下：

定义安全组

第一个字段为安全组名称。

sec.model：配置安全模式，它的可选值为 v1/v2c/usm

sec.name：设置安全体名称

定义视图

第一个字段为视图名。

incl/excl：配置所包括或者排除的 MIB 子树。

subtree：视图中所涉及的 MIB 子树。

mask：掩码

向安全组授权相应的视图

第一个字段为安全组名。

context：上下文，v1、v2c 中始终为空。

sec.model：安全模式，可选值为 v1、v2c、usm。

sec.level：安全级别，可选值为 auth、noauth、priv，v1 和 v2c 中只能为 noauth。

match：前缀，这个参数只有在中 v3 使用。他指定上下文如何与 PDU 中的上下文相匹配，

read：授权的读视图。

write：授权的写视图。

notif：授权的 trap 视图。

说明：对 snmpd.conf 的编辑只能用空格，不能用 Tab 键，否则会出错。每次改变 snmpd.conf 的内容后都必须重启 snmp 服务更改才能生效。

②启动 snmp

使用 snmpd 命令启动简单网络管理协议守护进程。

指令如下：

snmpd -c /etc/snmp/snmpd.conf –C

-c 指定配置文件路径，-C 表示代理只使用此配置文件。

③测试 snmp

启动 snmp 后可以使用以下指令测试 snmp 是否正常工作：

snmpwalk -v 2c -c public localhost

这条指令用于查看本机所有 SNMP 的信息，如果正确返回信息则说明 snmp 正常工作。

3.3 配置并编译内核，建立目标机测试环境

在第 2 章简单介绍了内核配置的选项，利用 ok6410 开发板的相关资料来进行开发，系统内核采用官方提供的，它已经配置好对应于开发板的相关信息，然而根据需要仍然要对 IPv6 和 IPv4 双协议栈进行配置。

3.3.1 配置 Linux 内核双协议栈支持

在提供的内核中 IPv4 协议栈已经支持，其实需要配置的是仅仅 IPv6 协议栈。首先进入内核源程序根目录，以 root 用户运行 make menuconfig 命令，然后可以找到[*] Networking support --->既网络支持选项。

选择这个选项并进入下一级选项卡，可以找到 Networking options --->选项

选择这个选项并进入下一级选项卡，可以找到 The IPv6 protocol ---->选项。

选择这个选项并进入下一级选项卡就可以配置 IPv6 协议了，如图 3-5 配置 IPv6 协议界面所示。

在图 3-5 中并不能显示所有的选项，可以查阅相关文档，根据的需要设置好 IPv6 协议的支持情况，然后退出，并保存内核设置。然后重新编译操作系统内核，系统就支持了 IPv6 协议。在下一节将介绍内核的编译。

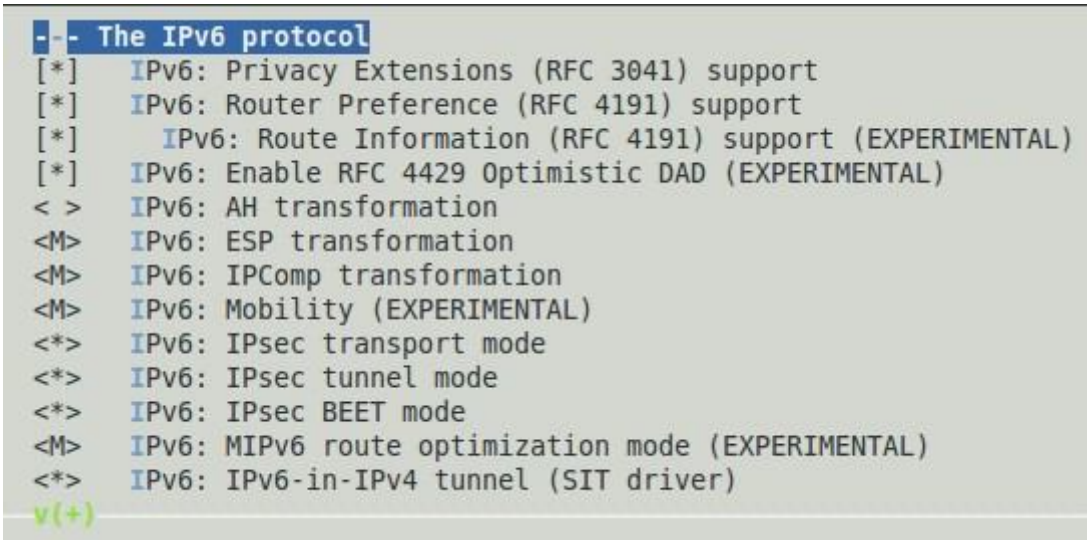

图 3-5 配置 IPv6 协议界面

Fig.3-5 Interface of configure IPv6

3.3.2 编译和移植 UBOOT、文件系统及 Linux 内核

根据开发嵌入式 linux 的需要，需要为目标板开发出 UBOOT、文件系统，这些东西的开发是一个巨大的工程，对本设计的开发没有多大意义，可以利用 ok6410 官方提供的 uboot 和文件系统，仅在必要的时候往里面添加相应的东西就可以了。

将 uboot 源码压缩包拷贝到 Ubuntu 的目录下，解压缩并编译，Ubuntu 下操作过程如下所示：

tar zxf uboot1.1.6_FORLINX_6410.tgz

进入 uboot1.1.6 目录、配置 config、编译：

cd uboot1.1.6（进入 uboot 源码的目录）

make smdk6410_config（配置 config）

make clean（删除以前编译时产生的文件）

make（编译）

编译如果成功，在“uboot1.1.6”目录下会产生一个的二进制文件“u-boot.bin”。该文件就是需要烧写到 Nandflash 的 U-boot 映像文件。

添加内容到文件系统的步骤：

S1、复制官方提供的 cramfsck，mkcramfs，Forlinx6410_touch_v1.0.cramfs 三个文件到 Ubuntu 系统。

S2、执行命令 sudo ./cramfsck -x myfs Forlinx6410_touch_v1.0.cramfs

该命令用于解压 Forlinx6410_touch_v1.0.cramfs 至 myfs 目录

S3、添加自己的内容至 myfs 文件夹内。

S4、执行 sudo ./mkcramfs myfs/ myfs.cramfs

该命令用于把 myfs 文件夹压缩成一个名为 myfs.cramfs 的 cramfs 文件镜像

编译 Linux-2.6.36.2 内核

在第 2 章内核配置过程中给出了内核配置命令，下面将给出内核配置和编译过程中的一些细节。

配置内核

将内核的压缩包‘FORLINX_linux-2.6.36.2.tar.gz’解压到工作目录下，命令如下：

tar zxf FORLINX_linux-2.6.36.2.tar.gz

FORLINX_linux-2.6.36.2.tar.gz 是 ok6410 公司提供内核压缩包，可以直接使用。将用‘make menuconfig’命令来配置内核，为了宿主机支持该命令，需要

安装‘libncurses5’，可以采用以下命令行来安装（使用这个命令需要宿主机连接到互联网）：

sudo apt-get install libncurses5-dev

编译内核

命令如下：

make zImage

编译成功后将在内核源码目录的 arch/arm/boot 中能够找到 Linux 内核映像文件：zImage

这一节产生的三个文件是最重要的：u-boot.bin、zImage、myfs.cramfs。将把这三个文件烧制到目标板上，形成本设计的最终产品。

3.4 编写温度传感器驱动

上面的嵌入式 linux 开发环境构建完成后，还是要再看一下的目标：就是管理连接到 ARM 板上的温度传感器数据，下面需要开发网管代理，同时需要代理与智能数据采集设备（传感器）之间能够进行数据交互，这里的数据交互需要另一个模块来完成这个功能。它既能够与代理交互把数据信息传到代理端 MIB 库中，又能够跟智能数据采集设备（温度传感器）交互，读、写相关数据，由于智能数据采集设备（温度传感器）是直接连接到 ARM 开发板上的，首先需要编写它的驱动程序。因此先要了解它的设计原理图。

3.4.1 温度传感器设计原理图

温度传感器 DS18B20 连接到 ARM11 上的原理图。如图 3-5 所示。GPE0 是 ARM11 的数据线端口。

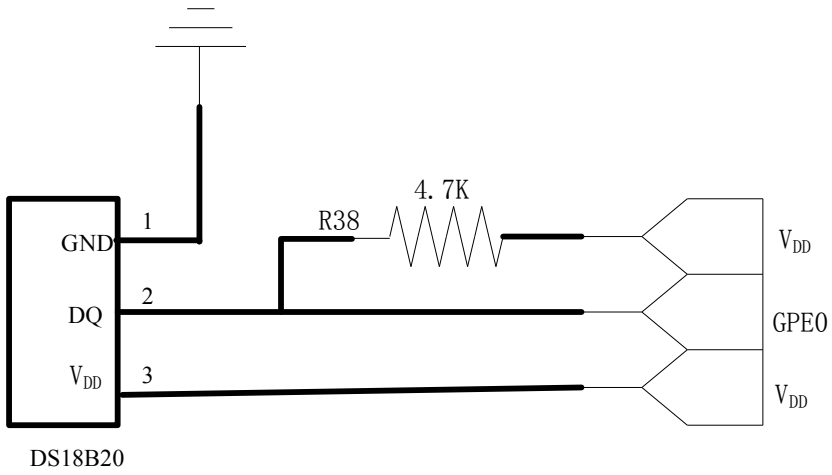

图 3-6 DS18B20 设计原理图

Fig.3-6 DS18B20 design schematics

3.4.2 编写温度传感器驱动程序

根据第 2 章对 DS18B20 的介绍，了解了它的很多特性，同时在第 2 章也简单介绍了它与控制器的交互流程，然而为了写驱动程序，首先要了解温度传感器更详细的工作原理，这是编写驱动程序的关键。下面详细的介绍 DS18B20 的工作原理。

3.4.2.1 温度传感器工作原理

如图 3-7 所示，DS18B20 与控制器（这里控制器是选择的 ARM11 芯片）交互工作周期。前两步是数据交互的“握手协议”，即：双方先要确认能够交互，后三步是进行数据交互。具体交互步骤如下：

S1、首先控制器将对 DS18B20 芯片复位。所谓复位就是指由控制器给连接有 DS18B20 上的单总线上加上至少 480uS 的低电平信号。DS18B20 接收到这个信号 15~60uS 后向控制器回发一个传感器芯片存在脉冲。在复位信号结束之后，控制器需要将数据单总线电平拉高，以等待接收在 15~60uS 后 DS18B20 发回的存在脉冲[32] 。

S2、存在脉冲是由 DS18B20 接收到复位信号后发回的信号，其实它是指一个 60~240us 的低电平信号。当 DS18B20 在指定的时间内发送完这个信号后，就和控制器之间达成了通信协议，接下来是控制器与 DS18B20 间的数据通信。复位低电平的时间长短和单总线的电路连接是否完好都会影响到存在脉冲的接收

是否正常，在驱动设计时需要注意对意外情况的处理。

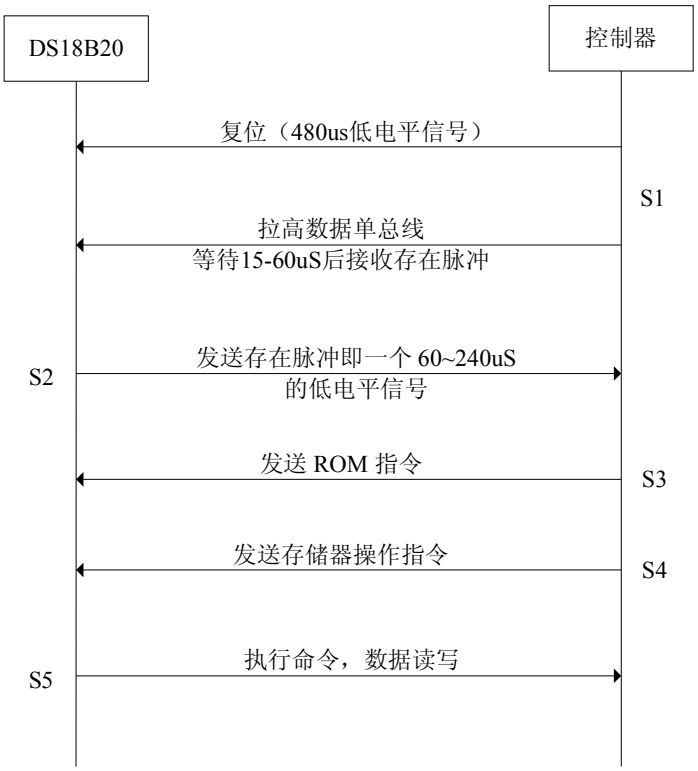

图 3-7 DS18B20 与控制器交互工作周期

Fig.3-7 Interaction cycle between DS18B20 and controller

S3、控制器发送 ROM 指令，双方“握手”完毕，就要进行“谈话”了。ROM 指令主要是对片内 ROM 进行操作的指令，长度是 8 位。ROM 指令有 5 条，分别是读 ROM 数据、指定匹配芯片、跳过 ROM、芯片搜索、报警芯片搜索。每一个工作周期只能发一条 ROM 指令。由于单总线上可以同时挂接多个器件，ROM 指令的主要目的是根据每个器件上所特有的 ID 号区分一条总线上挂接的多个温度传感器并作相应的处理。一般情况下，对于只挂接单个 DS18B20 芯片时只需要发送跳过 ROM 指令（这是特有的一条 ROM 指令而不是不发送 ROM 指令）。

S4、在发送完 ROM 指令之后，紧接着就是要发送存储器操作指令了。存储器操作指令是控制芯片的关键，它的作用是命令 DS18B20 做想要做的工作。控制器发送的存储器操作指令共 6 条分别是读 RAM 数据、写 RAM 数据、温度转换、将 RAM 数据复制到 EEPROM、将 EEPROM 中的报警值复制到 RAM、工作方式切换，存储器指令也是 8 位。

S5、当一个存储器操作指令发送结束后，DS18B20 将根据控制发送的指令

执行数据的处理操作即：执行命令，进行数据的处理。不同的存储器操作指令执行的操作方式不同，如果执行温度转换指令则控制器（ARM）必须等待 DS18B20 执行其指令后才能进行读温度的操作，一般温度转换时间为 500uS。

如果要读出当前的温度数据需要执行两次工作周期。

第一个周期：复位、跳过 ROM 指令、执行温度转换存储器操作指令、等待 500uS 温度转换时间。

第二个周期：复位、跳过 ROM 指令、执行读 RAM 的存储器操作指令、读数据。

其它的操作指令执行流程基本相同，在此不再多做介绍。

3.4.2.2 驱动编写

事实上，驱动的编写是个很复杂的事，首先要了解硬件的工作原理，其次要知道硬件与 ARM 是怎么样连接的，也就是说要找到硬件和 ARM 连接的接口是哪个，然后要知道内核的函数怎么调用和操作总线接口。也就是说，它的编写既要了解对应的硬件（如 DS18B20）、控制器 ARM，也要了操作系统内核 API 函数。上一节介绍了 DS18B20 的工作原理，下面在写驱动的时候就用到了。驱动的编写步骤概述如下：

S1、为设备申请设备号，使用的内核函数是：MKDEV

S2、注册设备，使用的内核函数是：register_chrdev_region

S3、设备信息的初始化，使用的内核函数是：cdev_init

S4、设备信息的添加，使用的内核函数是：cdev_add

S5、创建设备对应的设备类，使用的内核函数是：class_create 这一步的目的是为了能在内核 dev 目录下自动添加设备，否则需要 mknod 手动添加设备

S6、创建设备，使用的内核函数是：device_create 这一步就是在 dev 目录下自动添加设备

这只是设备初始化操作的具体步骤，一个设备的驱动程序还应该包括另外一部分“file_operations”，它就是把系统调用和驱动程序关联起来的关键数据结构，这个结构体函数实际上是一组操作函数指针的集合，这些操作是用作系统调用的操作，这个函数是内核进行定制的标准函数，需要按照这个标准函数进行驱动程序的编写。根据实际情况，并不需要实现所有结构体中的方法。通常，进行设备驱动程序的设计时主要注重如下几个方法的实现：

.owner = THIS_MODULE,

.llseek =***_llseek,

.read = ***_read,

.write = ***_write,

.ioctl = ***_ioctl,

.open = ***_open,

.release = ***_release,

在实现 read 和 write 方法的时候，需要对 ARM11 的 GPE0 口进行读写操作，用到 s3c_gpio_cfgpin(S3C64XX_GPE(0), S3C_GPIO_SFN(1)) 配置 GPE0 口为写模式、gpio_set_value(S3C64XX_GPE(0), 1)给 GPE0 口加高电平等函数，在此不一一列举解释了。

3.5 开发并移植 SNMP 代理

在代理的开发和移植的过程中，需要先在宿主机上将代理安装上去，然后测试成功后，再将代理交叉编译，移植到目标板。在第 2 章为了得出软件架构，已经简单介绍了代理开发的过程，下面是开发代理程序的详细过程。

3.5.1 设计 MIB 文档

MIB 是委托代理的关键，完善的 MIB 设计能降低协议处理的复杂性。要写一个 MIB 文档不难，但是要设计一个合理的 MIB 模块却并不容易，具体的模块要根据具体的需求来设计，对于一个模块是否需要用到表，对于一个表需要设计几个对象，应该给定对象什么样的访问权限等。这些都是要考虑的问题。在设计 MIB 的过程中，对于表，只有在代理管理的设备中，有多个同一类的设备，并且需要管理的数据是相同的情况下才用到。由于要管理的设备只是一个温度传感器，在此设计中没有用到表，只是设计三个用于管理的对象节点。只是做个相应的演示，没有具体的应用为依托，也不可能设计出有用的 MIB 文档。

在设计的 MIB 文档中，在 enterprises 下扩展定义了一个叫 hpu 的节点，oid 为 1.3.6.1.4.1.5959559，hpu 下又定义了三个叶子节点分别为 TmpFromSS、AlarmTmp、SensorNum。这三个节点分别记录传感器的温度，报警上限温度和传感器的序列号。这是记录传感器数据的节点。编写好 mib 文档并不能代理不能够直接识别这些文档，需要将它描述的节点信息，载入代理中，以便与网络上的

信息相互交互。

3.5.2 修改生成文件将 MIB 库文件载入代理

接下来，把 HPU-MIB.txt 复制到 NET-SNMP 的安装目录下，在我机器上需要的复制到目录/usr/local/share/snmp/mibs 下面。然后要在配置文件 snmp.conf 里面，把 HPU-MIB 库加进去。运行 snmpconf，就可以对 snmp.conf 进行设置，具体代码为+HPU-MIB。检查 mib 是否正常加载。命令如下：

snmptranslate -IR　-Tp hpu

如果加载正常，则显示如图 3-8 所示的树形图。

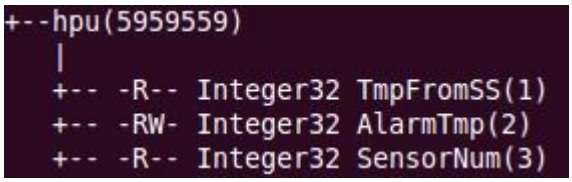

图 3-8 hpu 模块树形图

Fig.3-8 hpu module tree

通过模板生成.c 和.h 文件，运行命令：

mib2c -c mib2c.scalar.conf hpu

将在终端运行的当前文件夹内自动生成 hpu.c 和 hpu.h 文件，这两个文件是 scalar 模板生成的文件，但是需要修改 hpu.c 的代码才能够编译通过。

在 hpu.c 中添加自己的代码，定义的三个节点 TmpFromSS、AlarmTmp、SensorNum 分别需要设置默认值，并且根据管理器的需要收集温度传感器的信息，这里需要调用温度传感器的驱动来完成这些信息的收集工作，如果的驱动程序模块已经加载到的内核中，并且运行稳定，在此仅需要调用温度传感器驱动，读取温度传感器的温度、温度传感器的序列号，为 mib 节点提供可用信息。事实上，这个获取被管设备信息的代码模块就是代理-设备接口模块，它用来连接被管设备和代理程序，并进行数据的交换和更新。在多数情况下，它存在于被管设备中，是被管设备的一部分；然而它负责更新代理程序的数据。如果把代理程序看成是服务器的话，那么代理－设备接口模块就是为开发者提供一个客户程序开发平台，从逻辑上讲，它又是网管代理系统的一部分。

由于 TmpFromSS、SensorNum 是只读类型的节点，AlarmTmp 是可读可写的节点，对他们的处理过程不一样，对于只读类型的节点在处理过程与可读可写的

节点读部分的数据处理相同，在处理写部分时，过程相对复杂，如图 3-9 所示。

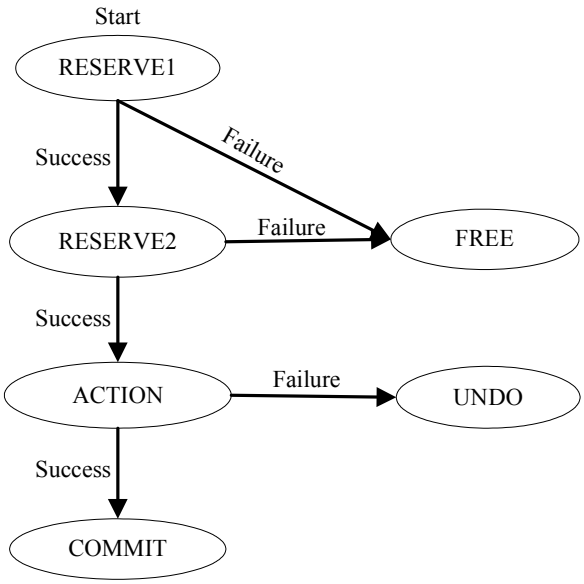

图 3-9 可写节点数据处理流程

Fig.3-9 Writable node Data processes

按照下列步骤对 NET-SNMP 重新编译：

S1、把 hpu.h 和 hpu.c 文件拷贝到 NET-SNMP 源文件下 agent/mibgroup 目录下；

S2、重新编译 NET-SNMP，运行命令 make；

S3、重新安装 NET-SNMP，运行命令 make install。

3.5.3 测试并移植代理

启动新编译安装的代理，使用命令：

snmpd -c /etc/snmp/snmpd.conf –C

然后在管理器端运行命令，查询或设置节点的值：

snmpget –v2c –c public IP 地址 TmpFromSS.0

snmpget –v2c –c public IP 地址 SensorNum.0

snmpget –v2c –c public IP 地址 AlarmTmp.0

snmpset –v2c –c rwusr IP 地址 AlarmTmp.0 i 40

经过测试可以对节点进行相应的读写。“IP 地址”处需要填写真实的代理所用的 IP。下面将要把这个代理移植到目标机上，在此需要用交叉编译环境，来重新编译的代理。配置命令如下：

./configure--with-cc=arm-linux-gcc--with-ar=arm-linux-ar--prefix=/usr/local/net-snmp-arm--build=i686-linux--host=arm-linux--with-endianness=little--disable-applications--disable-manuals--disable-debugging--disable-snmptrapd-subagent--disable-scripts--enable-mini-agent--with-mib-modules="notification"--with-mib-modules="agentx"--disable-ucd-snmp-compatibility--disable-perl-cc-checks--without-perl-modules--with-mib-modules="hpu"

说明：

--with-cc 指定采用的编译器

--with-ar 指定使用的编译库的工具

--prefix 安装目标目录

--build 指定在哪进行编译

--host 指定编译后运行的目标机类型

--with-endianness 配置目标机端大小

--disable-applications 不编译 SNMP 的应用

--disable-manuals 不编译帮助手册

--disable-debugging 不支持 debug

--disable-snmptrapd-subagent 不支持 snmp 陷阱子代理

--disable-scripts 不支持脚本

--enable-mini-agent 生成更小的代理

--with-mib-modules 指定需要编译的模块

--without-perl-modules 不支持 perl

然后编译和安装：

make

make install

配置 snmpd.conf

用 arm-linux-strip 裁剪文件大小，裁剪完后可以节省不少空间，这对于嵌入式有限的存储空间来说这么做是很有必要的。另外，在配置时加入--enable-mini-agent 选项生成的 snmpd 文件应该还能再小一些，然后将安装好的文件移植到目标机上，这个过程可以把这些文件直接放入目标机的文件系统，然后以系统文件的形式烧写到目标机中。移植完成后用同样的命令进行测试。

3.6 小节

本章为物联网终端确定了具体的系统组成，并设计了它的硬件结构和软件结构，详细描述了它的数据处理流程，通过使用 SNMP 网络管理模型来说实现对物联网终端的网络管理。通过构建嵌入式 linux 开发环境，安装配置 NET-SNMP，建立宿主机开发环境，配置编译内核，实现物联网终端的双协议栈，编写温度传感器驱动程序，开发了 SNMP 代理，在宿主机上测试，并移植到目标机上。为物联网终端实现及管理的开发形成了一个可行的方案。

4 总结与展望

4.1 总结

本文通过分析现阶段物联网发展现状，在做出未来物联网的实现是网络发展的未来趋势的判断的基础上，根据矿山信息化河南省高校工程技术研究中心针对物联网研究方面的需求，选择物联网终端网络管理实现研究为课题，做出了以物联网终端网络管理实现为目标的研究分析，通过对物联网终端相关硬件及软件技术的分析与研究，选择嵌入式 linux 操作系统作为实现此模块的系统平台，并以 SNMP 为管理平台，实现物联网终端的网络管理。所做的工作如下：

1 确定了物联网终端模块结构，分析研究了其所使用的硬件和软件，并在这个基础上，确立了物联网终端结构的基本组成。

2 根据物联网终端网络管理需要，拟定了它物联网终端的硬件结构以及相应的软件架构。

3 在研究的实现过程中，搭建了开发所需的系统环境、软件开发环境、交叉编译环境等，通过裁剪嵌入式 linux 内核，配置嵌入式系统对 IPv4/IPv6 双协议栈的支持。

4 为温度传感器 DS18B20 编写驱动程序。

5 根据物联网网络管理需求，开发针对物联网特定终端进行网络管理的代理程序。

6 移植代理程序到嵌入式开发板。

本研究是针对温度传感器终端进行的工作，在进行工作之初就把它当作一个案例，为以后的工作做出一个范例来。然而，物联网终端多种多样，涉及到的传感器技术也多种多样，使用 SNMP 来做网络管理所需要的 mib 库只能根据不同的传感器进行不同的设计，针对每一款传感器，都要在接入嵌入式开发板时编写相应的驱动，这些工作都会加大物联网终端管理实现的工作量。没有一个统一的标准，还没能够找到一个可以支持所有即使支持大多数终端的设计，都是物联网大规模应用的瓶颈。

4.2 展望

通过此次研究，使我找到了一种实现物联网终端及其管理的实现方案，同时

这个方案也面临着一些其他方面的技术挑战，比如本文采用的 SNMP 委托代理管理模型，在物联网中的应用时，由于物联网和计算机网络之间的不同，它的使用虽然能够解决网络管理的问题，然而，在使用它时仍然要针对物联网 的特点，将管理模型进行一些细节上的调整，使它能够更好的为物联网服务。也就是说需要针对物联网特点的网络管理模型进行新的研究，使其更好的服务于物联网应用。另外，在 SNMP 代理开发和 MIB 库建立过程中，开发过程漫长且复杂，这样的软件开发周期长，会影响到物联网的应用和推广，需要根据物联网网络管理等程序开发的特点，研究一整套的快速开发模型。

参考文献

[1] 于溥春. 浅谈物联网技术以及物联网技术在中国的发展 [J]. 硅谷, 2010,(13)

[2] 孙其博, 刘杰, 黎彝等. 物联网:概念、架构与关键技术研究综述 [J]. 北京邮电大学学报, 2010,33(3)

[3] 陈仲华. IPv6技术在物联网中的应用 [J]. 电信科学, 2010, 26(4)

[4] 董爱军, 何施, 易明. 物联网产业化发展现状与框架体系初探 [J]. 科技进步与对策, 2011, 28(14)

[5] 沈苏彬, 范曲立, 宗平等. 物联网的体系结构与相关技术研究 [J]. 南京邮电大学学报（自然科学版）, 2009, 29(6)

[6] 王文洋. 基于RFID技术的物联网探析 [J]. 科技信息, 2009, (26)

[7] European Research Projects on the Internet of Things(CERP-IoT)Strategic Research Agenda(SRA). Inter-net of things—strategic research roadmap［EB/OL］(2009).

[8] Commission of the European communities, Internet ofThings in 2020, EPoSS, Brussels［EB/OL］. (2008).

[9] 温家宝. 2010年政府工作报告［EB/OL］. (2010-03-15)

[10] 于泳, 贾会迎. 物联网关键技术及应用 [J]. 知识经济, 2011, (11)

[11] 刘化君. 物联网关键技术研究 [J]. 计算机时代, 2010, (7)

[12] 刘强, 崔莉, 陈海明. 物联网关键技术与应用 [J]. 计算机科学, 2010, 37(6)

[13] 张绍钧. 物联网关键技术及其应用 [J] . 信息化研究, 2011, 4

[14] 吕睿超. 基于异构网络融合的WSN网关设计与实现 [D]. 2010.

[15] 曹岑. 基于ARM的嵌入式智能网络仪表研究 [D]. 2009.

[16] 郭朗. 基于ARM的嵌入式系统设计 [D]. 2007.

[17] 倪礼君. 嵌入式平台上实时性Linux的裁减与实现 [D]. 2007.

[18] 肖贺. 基于OFDM的嵌入式无线终端 [D]. 2009.

[19] 宫晓华. 基于ARM的嵌入式系统设计 [D]. 2007.

[20] 崔健. 基于ARM的喷绘设备控制系统的设计与实现 [D]. 2010.

[21] 彭剑辉. 基于SNMP的多子网物理拓扑发现方法的研究与实现 [D]. 2009.

[22] 黄明. 基于SNMP和JAVA的网络性能管理 [J]. 黑龙江科技信息, 2008, (35)

[23] 印永强. HFC网络系统管理 [J]. 有线电视技术, 2008, 15(5)

[24] 徐丽, 封红旗. 基于SNMP协议的网管系统的研究与设计 [J]. 常州大学学报(自然科学版), 2010, 22(4)

[25] 罗雅过. 基于SNMP的MIB库访问实现研究 [J]. 西安文理学院学报（自然科学版）, 2010, 13(4)

[26] 赵定远. 嵌入式技术发展及产业链结构 [J]. 成都大学学报（自然科学版）, 2009, 28(1)

[27] 李国晓. 嵌入式系统在工业控制中的应用及发展趋势 [J]. 中国西部科技, 2011, 10(13)

[28] 刘佳佳. 浅谈μ CLinux与标准Linux在内存管理上的区别 [J]. 科技信息, 2009, (19)

[29] 俞海平. IPv6过渡技术在校园网升级中的应用 [J]. 农业网络信息, 2010, (5)

[30] 雷鸿. 基于虚拟机架构下嵌入式开发环境搭建的研究与实现 [J] . 信息通信, 2011, (4)

[31] 付强. 基于NET-SNMP的代理扩展开发 [J] . 工业控制计算机, 2011, (24)

[32] 朱丽丽，王长友. 基于AVR单片机与温度传感器DS18B20的多点温度测量 [J]. 电工电气, 2010, (12)

作者简历

一、基本情况

姓名：孟千胜 性别：男 民族：汉 出生年月：1984-03-14 籍贯：河南省漯河市召陵区

2005.09—2009.06 本科就读于河南理工大学计算机科学与技术学院，计算机科学与技术专业。

2009.09—2012.06 研究生就读于河南理工大学计算机科学与技术学院，计算机应用专业。

二、学术论文

1. Yang, Lishen, Meng, Qiansheng. Implementation of SNMP-based network management system's MIB for dual-stack. [C] 2011 International Conference on Electrical Information and Mechatronics（ICEIM2011）.

三、获奖情况

1. 2009-2010 年度，河南理工大学，三等奖学金、优秀研究生干部。
2. 2010-2011 年度，河南理工大学，二等奖学金、优秀研究生干部。
3. 2011 年 11 月，河南理工大学，十佳研究生干部。

学位论文数据集

关键词*	密级*	中图分类号*	UDC	论文资助
物联网;嵌入式技术;简单网络管理协议;代理;温度传感器;网络管理	公开	TP317.4	620	创新基金

学位授予单位名称*	学位授予单位代码*	学位类别*	学位级别*
河南理工大学	10460	工学	硕士

论文题名*	并列题名*	论文语种*
物联网终端网络管理研究与实现	Research and Implementation of the network management of Internet of Things	中文

作者姓名*	孟千胜	学号*	210909010024

培养单位名称*	培养单位代码*	培养单位地址	邮编
河南理工大学	10460	河南省焦作市	454000

学科专业*	研究方向*	学制*	学位授予年*
计算机应用	计算机网络	3	2012

论文提交日期*	2012.04

导师姓名*	杨立身	职称*	教授

评阅人	答辩委员会主席*	答辩委员会成员
	贾宗璞	

电子版论文提交格式　文本（　）　图像（　）　视频（　）　音频（　）

多媒体（　）其他（　）

推荐格式：Microsoft Word(DOC)；Adobe Reader（PDF）

电子版论文出版（发布）者	电子版论文出版（发布）地	权限声明

论文总页数*	59

注：共 33 项，其中带*为必填数据，为 22 项。