分类号 TP UDC 39

重庆大学

硕士学位论文

论文题目：基于 TinyOS 的无线电子鼻传感器网络节点研究

论文作者：杨红远

专　　业：控制理论与控制工程

论文导师：柴　毅　教授

答辩日期：2008 年 6 月 2 日

基于 TinyOS 的无线电子鼻传感器网络节点研究

重庆大学硕士学位论文

学生姓名：杨红远

指导教师：柴　毅　教授

专　　业：控制理论与控制工程

学科门类：工 学

重庆大学自动化学院

二 OO 八年四月

Reserch on Wireless E-Nose Sensor Network Nodes Based on TinyOS

A Thesis Submitted to Chongqing University
in Partial Fulfillment of the Requirement for the
Degree of Master of Engineering
by

Yang Hongyuan

Supervisor: Prof. Chai Yi

Major:Control Theroy and Control Engineering

College of Automation of Chongqing University ,
Chongqing, China.

April,2008

摘　要

无线传感器网络就是由部署在监测区域内大量的廉价微型传感器节点组成，通过无线通信方式形成一个多跳的自组织网络系统，其目的就是协作的感知、采集和处理网络覆盖区域中感知的对象，并发送给观察者。论文综述了无线传感器网络、电子鼻的国内外发展现状，讨论了无线传感器网络实现的相关理论及关键技术、传感器节点技术以及 IEEE 802.15.4/ZigBee 协议标准，主要对无线传感器网络节点在硬件、软件开发工具 TinyOS 以及具体的软件设计方面进行了深入研究，并完成基于气敏传感器的无线电子鼻传感器节点是实现。

论文在传感器节点硬件设计上进行了模块化电路设计，重点讨论了基于 MSP430F149、ATmega128L 的处理器模块设计，CC2420 的无线通信模块设计，详述了 CC2420 与各处理器之间接口引脚使用。针对于 Sink 传感器节点接入接口探讨了 RS232 和 USB 两种接口设计方案，其中通过 FT2232C 芯片实现串口到 USB 接口转换，简述了电源供应模块和传感器供电开关切换应用电路，最终以 ATmega128L+CC2420 进行了传感器节点 PCB 设计，完成传感器节点在硬件设计上的实现。

论文在进行传感器节点的软件设计之前首先研究了源代码开放的专用于无线传感器网络开发的微型嵌入式操作系统 TinyOS，探究了 TinyOS 操作系统组件模型、通信模型、事件驱动机制与并发模型以及内核的调度机制和策略以及不足。深入了解 TinyOS 操作系统是进行良好传感器节点程序设计的基础。针对具体的无线电子鼻程序设计详细分析了 TinyOS 程序设计 mian 函数入口、基于 TinyOS 的几种传感器节点消息数据包结构，具体的分析了传感器节点无线通信模块无线收发流程和串口通信模块、ADC 模块程序实现过程。研究了基于 TinyOS 的多跳组件，详述了基于最小跳数的路由算法，分析了传感器节点在路由过程中链接质量估计、邻居节点数据表管理的实现和问题，以及具体实现多跳通信协议执行框架中父节点选择、循环回路探测、重复数据包剔除等问题，并在此基础上实现基于最小跳数算法的节点多跳通信。

论文最后完成了基于气敏传感器的无线电子鼻传感器节点实现，并对测试结果进行了分析。通过测试表明，传感器节点在硬件方面、软件方面各个模块工作稳定，电子鼻节点功能正常。

关键词：传感器节点，电子鼻，TinyOS，多跳通信

ABSTRACT

Wireless sensor networks (WSN) consists of a large number of low-cost micro-sensor nodes,which make up of a muliti-hop self-organizing network through wireless communication.Its purpose is to send sampling data in the network coverage region to observers.The paper has summarized the development of wireless sensor networks、E-nose at home and abroad and discussed the related theories and key technologies,sensor nodes and IEEE 802.15.4/ZigBe Protocol standards,and mainly done the intensive study of wireless sensor nodes in the network hardware,software development tools TinyOS,software design aspects,and according to actual needs E-nose wireless sensor nodes are realized based on the gas sensors.

The paper carried out modular design in the sensor node's hardware design.It put emphasis on processor module design based on the MSP430F149、ATmega128L processors,wireless communications module design based on CC2420 chip,and detailed the CC2420 pin interface between the processor used.Against Sink node access interface design with twoRS232 and USB interface which uses FT2232C chip to convert uart to USB interface.and compendiumed the design of power module and sensor board module. Utimately using ATmega128L+CC2420 design proposal to PCB, achieved node's hardware design.

Before node's software design,the paper in aspect of node's software design firstly researched micro- embedded TinyOS operating system which is an open source code operating system specialed for wireless sensor network.It has explored TinyOS's system components model,communication model, event-driven scheduling mechanisms, as well as the core mechanism and strategies. TinyOS is a thorough understanding of good design procedures based nodes.Next it did detailed analysis of the data structure TinyOS node information,and carried out wireless transceivers, serial communication module and ADC module program design and at last did the tatal analysis of TinyOS's multi-hop communications components,recounted minimum hop routing algorithm,as well as link quality estimate and neighborhood table management,and father selection,cycles,duplicate packet elimination,queue management in achieve specific Multi-hop communication framework process.On this basis, finish the multi-hop communications based on minimum hop routing algorithm.

At last,the paper achieved the design of E-nose sensor nodes based on gas

sensors,and the test results were analyzed. By the test, the sensor nodes in the hardware modules work stability, and in this sensor node hardware platform software modules operating normally.

Keywords: Sensor node, E-nose,TinyOS, Multi-hop Communition

目　　录

1 绪　论

1.1 问题的提出

20 世纪 60 年代以来，环境问题逐渐从地区性问题演变为全球性问题。随着石油、化工、钢铁等产业的发展和大量易燃、易爆、有毒、有害气体的出现，为保障生产的安全，工人健康或检验产品的质量，基于气敏传感器采样、传输、处理、识别功能于一身的电子鼻系统应运而生。电子鼻系统是一种智能仿生传感器系统，能够准确的检测识别出各种简单和复杂有毒气体。但是传统的电子鼻系统只能针对工作在某些特定空间的环境下小空间范围内的气体检测，不适用于生产现场、厂矿地区、边远野外等大范围场合应用。

随着半导体技术和通信技术的发展，传统 PC 时代和网络时代正在向普适计算时代迈进。“普适计算”是指“无论何时何地，人们都可以通过某种设备访问到所需要的信息”[1]。信息设备还可以非常廉价的通过无线网络和互联网连接，并可根据用户的个性需求进行定制，以嵌入式商品的方式呈现在人们的工作和生活中，甚至是以与人们日常生活中经常碰到的器具融合在一起的多样形式体现[2][3]。普适计算一个重要研究就是无线传感器网络技术。无线传感器网络就是由部署在监测区域内大量的廉价微型传感器节点组成，通过无线通信方式形成一个多跳的自组织的网络系统，能够协作的感知、采集和处理网络覆盖区域中感知对象的信息，并发送给观察者[4]。一种无线电子鼻传感器节点研制的提出使得针对各种复杂、大范围环境的气体检测成为可能，促使环境监测系统向网络化、数字化、区域化、全球化、实时性、综合性、信息形式多样化发展趋势发展。

1.2 无线传感器网络研究综述

1.2.1 无线传感器网络概述

传感器技术、通信技术和计算机技术是现代信息技术的三大支柱，它们分别完成对被测量对象的信息提取、信息传输及信息处理。随着它们的飞速发展，具有感知、计算存储和通信能力的微型传感器开始出现在军事、工业、农业和宇航各个领域。这些微型传感器具有无线通信、数据采集和处理、协同合作等功能，无线传感器网络应用而生。

传感器网络结构如图 1.1 所示。传感器节点在网络中可以充当数据采集者、数据中转或类头节点的角色。作为数据的采集者，数据采集模块收集周围环境的数据（温度、湿度、声音），处理单元通过网络将数据传输到远方基站或 Sink 节点；作为数据中转站，节点除了具备数据采集功能外，还要具有接收临近节点采集的

数据能力，经数据融合后，发送到基站或汇聚节点。采集数据通过互联网或卫星等途径传输给用户。

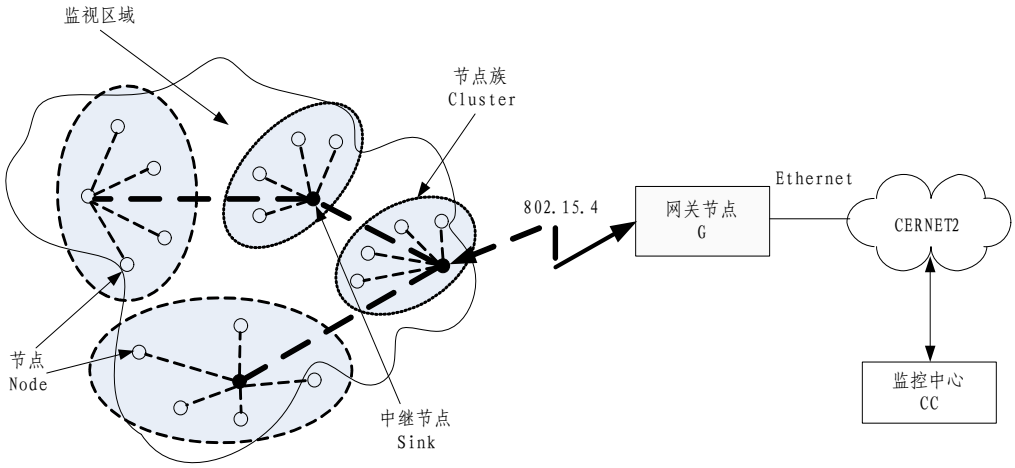

图 1.1 传感器网络结构

Fig.1.1 Sensor network structure

相对于传统无线网络，无线传感器网络具有一些明显的特征：

① 网络中传感器节点数量庞大、分布密集，单位面积所拥有的网络节点数远大于传统的无线网络。

② 网络中节点采用自组网的通信方式。传感器节点是自主的、自治的，节点之间以 Ad Hoc 方式通信，不同于传统无线网络中心的控制模式。

③ 网络的拓扑易变化。由于环境影响和节点能量有限，节点容易出现故障，导致网络拓扑信息变化快速。

④ 传感器节点能量、处理能力、存储能力、通信能力等十分有限。通常情况下，传统无线网络的首要设计目标是要提高服务质量和高效带宽利用，其次才考虑节约能量，而传感器网络首要设计目标就是能量的高效利用。

⑤ 网络应具备较强容错能力。传感器网络很容易受节点能量变化、计算存储能力、所处自然环境恶劣的影响，导致网络出现故障，易出错，这要求无线传感器网络具有较强的容错能力，才更具有应用前景。

⑥ 以数据为中心的网络。传感器网络是一种任务型网络，用户可以通过查询事件方式，直接将所关心的事件通告给网络，而不是通告给某个确定的网络节点，网络在获得指定事件的信息后报告给用户。这是一种以数据本身作为查询或传输线索的思想。

1.2.2 无线传感器网络节点

传感器节点是组成无线传感器网络的基本单位，是构成无线传感器网络的基础平台。传感器节点具有信息采集、融合并传送数据的功能，节点中的电源模块负责节点的驱动，是决定网络生存期的关键因素。

传感器节点通常是一个微型嵌入式系统，它的处理能力、存储能力和通信能力相对较弱，通过携带能量有限的电池供电。传感器节点一般由传感器模块、处理器模块、无线通信模块和能量供应模块四部分组成。如图 1.2 所示。传感器模块负责监测区域内信息的采集和数据转换；处理器模块负责控制整个传感器节点的操作，存储和处理本身的数据及其它节点发送来的数据；无线通信模块负责与其它传感器节点进行无线通信，交换控制消息和收发采集数据；能量模块为传感器节点提供运行所需的能量。

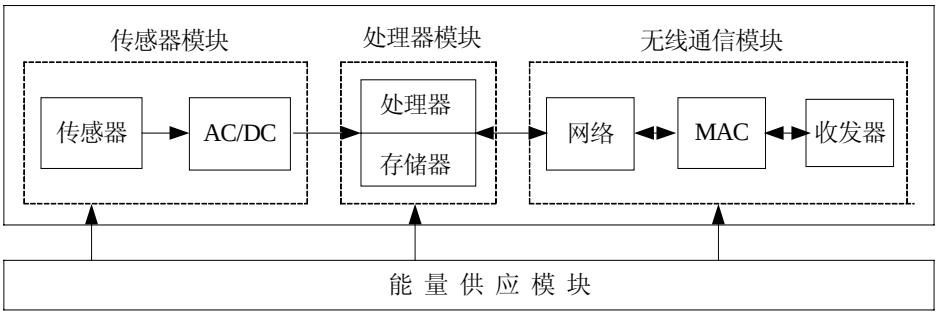

图 1.2 无线传感器节点体系结构

Fig.1.2 Wireless sensor node structure

在传感器节点设计中一般需要从以下几个方面考虑[4]：

① 微型化

无线传感器网络节点应该在体积上足够小，保证对目标系统本身的特性不会受到影响。在软件设计方面，要求所有模块的软件都应尽量精简，没有冗余代码。

② 稳定性和安全性

无线传感器节点的硬件稳定性要求节点的各个部件都能够在给定外部环境条件变化范围内正常工作。一方面系统在各种恶劣的气候条件下不会损坏；另一方面所有测量探头都能够尽量接近检测环境以获得最真实的参数信息。节点的稳定性还要求在软件方面得到保证。一方面，软件模块要保证其逻辑上的正确性和完整性；另一方面在硬件出现问题的时候能够及时感知并采取积极的措施。另外，对于敏感数据要以密文的形式存储和发送，并要有数据完整性保护，以防治外界因素造成数据的修改。

③ 扩展性和灵活性

无线传感器节点需要定义统一、完整的外部接口。在需要添加新的硬件部件时可以在现有节点上直接添加，而不需要开发新的节点，同时，节点可以按照功能拆分成多个组件，组件之间通过标准接口组合。在软件方面不需要额外的设备就可以自动升级，软件模块要做到组件化和可配置。

④ 低成本

低成本是传感器节点的基本要求，只有低成本，才能大量的布置在目标区域中，表现出无线传感器网络的优点，同时低成本对传感器的各个部件也提出了苛刻的要求。

另外，无线传感器网络节点在实现各种应用系统的同时存在电源能量有限、通信能力有限、计算和存储能力有限等现实约束。现有的典型的节点有 Berkeley Motes，Sensoria WINS，Berkeley PiconodeS，SmartMesh Dust mote 等。

1.2.3 IEEE802.15.4/**ZigBee** 协议标准

IEEE 802.15.4/ZigBee 协议满足国际标准组织(ISO)开放系统互连(OSI)参考模式。它定义了单一的 MAC 层和多样的物理层，如图 1.3。ZigBee 联盟制定了 MAC 层以上协议，其协议套件由高层应用规范、应用会聚层、网络层、数据链路层和物理层组成[6]。

<table>
<tr><td colspan="2">ZigBee Profiles</td></tr>
<tr><td colspan="2">网络应用层</td></tr>
<tr><td colspan="2">数据链路层</td></tr>
<tr><td>IEEE802.15.4 LLC</td><td>802.2 LLC</td></tr>
<tr><td colspan="2">IEEE802.15.4 MAC</td></tr>
<tr><td>868/915 PHY</td><td>PHY2400 PHY</td></tr>
</table>

图 1.3 802.15.4 协议架构

Fig.1.3 802.15.4 protocal structure

物理层：IEEE 802.15.4/ZigBee 定义了 2.4GHz 物理层和 868/915MHz 物理层两个物理层标准，它们都采用了 DSSS（直接序列扩频）。

2.4GHz 波段为全球统一的无需申请的 ISM 频段，有助于设备的推广和生产成本的降低。2.4GHz 物理层通过采用高阶调制技术能够提供 250kbps 的传输速率，有助于获得更高的吞吐量、更小的通信时延和更短的工作周期，从而更加省电[6]。

在 IEEE 802.15.4/ZigBee 中，总共分配了 27 个具有三种速率的信道：在 2.4GHz 频段有 16 个速率为 250kbps 的信道，在 915MHz 频段有 10 个 40kbps 的信道，在 868MHz 频段有 1 个 20kbps 的信道。一个 IEEE 802.15.4 网可以根据可用性、拥挤状况和数据速率在 27 个信道中选择一个工作信道。从能量和成本效率来看，不同

的数据速率能为不同的应用提供较好的选择。

MAC 层：IEEE 802 系列标准把数据链路层分成 LLC（逻辑链路控制）和 MAC（媒介接入控制）两个子层。LLC 子层在 IEEE 802.2 标准中定义，为 802 标准系列共用；而 MAC 子层协议则依赖于各自的物理层。IEEE 802.15.4 的 MAC 协议包括以下功能：设备间无线链路的建立、维护和结束；确认模式的帧传送与接收；信道接入控制；帧校验；预留时隙管理；广播信息管理。MAC 子层提供两个服务与高层联系，即通过两个服务访问点（SAP)访问高层。通过 MAC 通用部分子层 SAP（MCPS-SAP）访问 MAC 数据服务，用 MAC 层管理实体 SAP（MLME-SAP）访问 MAC 管理服务。这两个服务为网络层和物理层提供了一个接口[5]。灵活的 MAC 帧结构适应了不同的应用及网络拓扑的需要，同时也保证了协议的简洁。

IEEE 802.15.4/ZigBee 帧结构的设计原则为保证网络在有噪音的信道上能够足够健壮性的传输的基础上将网络的复杂性降到最低。每一后继的协议层都是在其前一层添加或者剥除了帧头和帧尾而形成。IEEE 802.15.4/ZigBee 的 MAC 层定义了 4 种帧结构：信用帧、数据帧、响应帧和 MAC 命令帧。另外，IEEE 802.15.4/ZigBee 标准，即 LR-WPAN(低数据率的无线个人网)标准还支持可选的超帧结构。该超帧结构的格式是由协商者来定义，绑定了网络信标帧，而由协商者来使用的。超帧被划分为 16 个 PHY 层大小相等的时隙，信标帧在每一个超帧的第一个时隙中进行传输，如果协商者不希望使用超帧结构，它可以关掉信标帧的传输，信标帧可以用来同步网络中的设备，识别 PAN 并且描述超帧结构。在冲突访问阶段，任何一个设备如果想进行通信，必须与其它设备使用 CSMA-CA 的机制，而且所有的事务必须在下一个网络信标帧到来前完成[7]。

对于低延迟或者有特殊数据带宽要求的应用，PAN 的协商者可以利用部分活动的超帧结构来做到，它们被称为确保服务的同步时隙（GTS），这些时隙是由信道无竞争周期（CFP）组成。CFP 一般出现在活动的超帧尾端，前面一般跟随着一些信道竞争访问周期（CAP）。一个 PAN 协商者可能包括少于 7 个的 GTS，而每个 GTS 一般占用不止一个时隙。但是协商者中还必须保留一定的时隙作为其他网络设备访问或者一个新的设备访问该网络进行相应的通信之用，但是在 CFP 到来之前所有基于竞争的事务必须全部完成，而且传递 GTS 的每个设备必须保证其事务在下一个 GTS 到来之前和当前 GTS 的 CFP 结束之前完成。正是由于标准中定义的这种超帧结构，才保证了该协议具有极低的功耗特性。

IEEE 802.15.4/ZigBee 协议使用 MAC 层的安全机制，来保证 MAC 命令帧、信标帧和响应帧的安全性。单跳的数据消息是通过对 MAC 层的安全来做到的，而多跳的消息报文一般是通过更上层(如网络层)的安全机制来保证的。ZigBee 的 MAC 层使用了一种被称为高级加密标准（AES）的算法进行加密的，并且它基于

AES 算法生成一系列的安全机制，用来保证 MAC 层帧的机密性、一致性和真实性[7][8]。

1.3 无线传感器网络与电子鼻研究现状

1.3.1 无线传感器网络研究现状

2003 年 2 月份的美国《技术评论》杂志(《Technology Review》)评出对人类未来生活产生深远影响的十大新兴技术，传感器网络列为第一。无线传感器网络被认为是 21 世纪最重要的技术之一，它将会对人类未来的生活方式产生深远影响。

1998 年，UCLA 和 Rockwell 研究中心在 DARPA 支持下进行的 WISN (Wireless Integrated Sensor Network)课题研究是传感器网络研究的开端，其目的主要是通过嵌入式仪器、设备和环境中的传感器、执行机构和处理器构建一个分布式网络环境，提供 Internet 的访问能力[4]。

从 2000 年开始，无线传感器网络因其巨大的应用价值，已经引起了世界许多国家的军事部门、工业界和学术界的极大关注。美国和欧洲相继启动了许多关于无线传感器网络的研究计划。2003 年美国自然科学基金委员会制定了无线传感器网络研究计划，在加州大学洛杉矶分校成立了传感器网络研究中心，联合周边的加州大学伯克利分校、南加州大学等展开了“嵌入式智能传感器”研究项目。鉴于无线传感器网络涉及多学科交叉的研究领域，具有鲜明的跨学科研究的特点，美国所有著名高校和加拿大、英国、德国、芬兰、日本、意大利等国也都加入从事传感器网络相关技术的研究。

我国在传感器网络方面的研究工作相对发展较慢，目前国内一些高等院校与研究机构已经积极开展无线传感器网络的相关研究工作，主要有中科院、哈尔滨工业大学、清华大学、北京邮电大学、西北工业大学、天津大学和国防科技大学等。同时有关无线传感器网络的一些相关产品也相继脱颖而出，极大的推动了国内传感器网络研究的发展[4]。

无线传感器网络已经成为当今信息领域的研究热点，在其关键技术方面已取得不少成果：在网络拓扑控制方面：拓扑控制可以分为节点功率控制和层次型拓扑结构形成两个方面。在功率控制机制方面，提出了 COMPOW 等统一功率分配算法，LINT 和 LMN/LMA 等基本节点度数的算法，CBTC、LMST、RNG、DLSS 等基本邻近图的近似算法。对于层次型的拓扑控制目前提出了 TopDisc 成族算法、改进的 GAF 虚拟地理网格分族算法，以及 LEACH 和 HEED 等自成族算法[14]。另外除了传统的功率控制和层次行拓扑控制，还提出了启发式的节点唤醒和休眠机制。在网络协议方面：目前研究的重点是网络层协议和数据链路层协议，已经提出了多种类型的传感器网络路由协议：多个能量感知的路由协议，定向扩散和谣

传路由等基于查询的路由协议，GEAR 和 GEM 等基于地理位置的路由协议，SPEED 和 ReInForM 等支持 QoS 的路由协议，其中关于 MAC 协议提出了 S-MAC、T-MAC、和 SIFT 等基于竞争的 MAC 协议，DEANA、TRAMA、DMAC 和周期性调度等时分复用的 MAC 协议，以及 CSMA/CA 与 CDMA 相结合、TDMA 和 FMDA 相结合的 MAC[15]。在网络安全方面：SPINS 安全框架在机密性、点到点的消息认证、完整性鉴别、新鲜性、认证广播方面定义了完整有效机制和算法，其中随机密钥对模型、基于多项式的密钥对模型是当前最有代表性的算法[26] 。在时间同步方面：已经提出了多个时间同步机制，其中 RBS、TINY/MINI-SYNC 和 TPSN 被认为是三个基本的同步机制。在定位技术方面：通常会使用三边测量法、三角测量法或极大似然估计法确定节点位置，目前提出的定位机制主要有质心算法、DV-Hop 算法、Amorphous 算法、APIT 算法。在数据管理方面：美国加州大学伯克利的分校 TinyDB 系统和 Cornell 大学的 Cougar 系统是目前具有代表性的传感器网络数据管理系统，传感器网络的数据管理系统的结构主要有集中式、半分布式、分布式及层次式结构，目前大多数研究工作集中半分布式结构方面。无线通信技术方面：IEEE802.15.4 协议标准是很多无线传感器网络的无线通信平台，超宽带技术（UWB）是一种极具潜力的无线通信技术。超宽带技术具有对信道衰落不敏感、发射信号功率谱密度低、低截获能力、系统复用度低。迄今为止关于 UWB 有两种技术方案，一种是 DS-CDMA 单频带方式，一种是多频带 OFDM 方案，但还没有一种方案成为国际标准；在嵌入式操作系统方面：目前在科研机构的研究中广泛使用美国加州大学伯克利分校针对无线传感器网络研发的 TinyOS 操作系统，但仍然存在不足之处[4]。

近年来，无线传感器网络取得了不少的研究成果，对于无线传感器网络的关键技术有了新的深入，相继出现了一些新的演示系统，例如 Smart Dust 利用 MEMS 技术设计微型化的传感器节点，总体积约 100 立方毫米的节点可以像尘埃一样悬浮在空气之中，实现阶段在实际商务中使用来自传感器网络的感知数据 UCBerkeley 的 Marco Motes 节点配置了温度、湿度、压力、声音、图像、磁场等多种传感器在目标跟踪系统、环境监测系统中应用的十分成功。我国中科院计算所的 Gaints 节点系列已经逐步开始成熟。另外我国的哈尔滨工业大学在传感器数据管理系统方面，提出了以数据为中心的传感器网络的数据模型、一系列的能源有效的感知数据操作算法和感知数据操作算法和感知数据查询处理技术，并研制了传感器网络数据库管理系统[24]。

1.3.2 电子鼻研究现状

客观需求推动了电子鼻技术的发展。人和动物的嗅觉在人类的生产和生活中一直扮演着重要角色，很早以前人们就利用警犬异常灵敏的嗅觉来识别犯罪嫌疑

人留在现场的气味，帮助破案。随着社会的发展，嗅觉在生活中的重要性有所下降，但在某些特殊领域却显得格外重要，有些场合并不适合人和动物直接接触，如环境污染监测、有毒气体检测等，因此人们致力于人工嗅觉，即电子鼻的研究。早在 1962 年，Taguch 和 Seiyama 就分别报道了用金属氧化物检测还原性气体的工作，1968 年推出第一个商品化的用于检测室内气体泄漏的氧化锡气体传感器(TGS 系列)[9]。但 20 世纪 60 年代研制的“电子鼻”并非现代意义上的嗅觉模拟系统。直到 20 世纪 80 年代初，作为气味分类用的新型智能传感器的概念才真正被提出。1982 年，英国 Warwick 大学的 Persaud 和 Dodd 用 3 个商品化的 S_nO_2 气敏传感器(TGS813，812，711)模拟生物嗅感受器细胞，并对乙醇、乙醚、戊酸等有机挥发气体进行了类别分析[10]。从那时起，人们不断探索用嗅觉模拟系统测定简单气体的类别和浓度，涉及领域十分广泛。20 世纪 80 年代末期，嗅觉模拟研究进入了快速发展时期。电子鼻主要应用在以下几个方面：

①食品工业

仅用一个石英振荡晶体传感器就可对加热蒸发后的红葡萄酒、白葡萄酒和玫瑰红葡萄酒进行分类识别。通过对传感器的响应曲线提取九种特征值，用主成分分析方法和三层前向神经网络可以 100%区分上述三种葡萄酒。利用 WO3 薄膜传感器阵列，采用主成分分析和聚类分析作为模式识别方法，在高温下对橄榄油、植物油、水果汁、番茄酱和其它饮料等进行识别研究，结果较好。

法国和日本的科学家们对食品新鲜度的判别问题，特别是肉类新鲜度，进行了积极探索[11]。用电子鼻技术可以很好地测定食品的气味质量，由此可判断食品新鲜程度，并且能对食品腐烂的时间进行估计。

②环境监测

环境监测主要是集中在对微量有害有毒气体的检测和监测上。文献[9]利用微型 S_nO_2 传感器阵列对含有微量 CO、CH4 和乙醇的空气进行了识别研究。文献[12]利用德国耶那环境传感器技术股份有限公司生产的 GGS1000-GGS7000 气体传感器对室内空气质量进行监测。致癌物质如苯和甲苯的监测也引起了人们的高度重视。

③医学诊断

早在 1995 年 John slater 等就利用电子鼻进行了与肾有关的疾病的诊断研究。目前电子鼻多用于糖尿病、与肾有关的疾病和一些细菌的类型和生长阶段的识别方面的诊断。意大利科学家最近研制出一种电子鼻，它可嗅出人体各种疾病的气味，是早期发现疾病的一种有力武器[12]。这种电子鼻不会受外部干扰，只要向其配备的气球内吹一口气，电子鼻即可通过生物传感器捕捉到其中含有的各种气味，经过计算机处理后形成化学图谱，协助医生分析患者的健康状况。这种电子鼻还

可以嗅到人体皮肤各种腺体发出的气味，以及皮肤寄生虫和细菌发出的气味。所以，对各种疾病都可以通过气味图谱进行诊断，它比现在的诊断方法简便得多。随着材料学和制作工艺水平的不断提高，经过英国、法国、德国等大学研究人员十几二十年的努力，电子鼻已经达到了相当水平。

相对而言，我国在电子鼻方面的研究起步较晚，主要研究单位有西安交通大学、中科院电子学研究所、中科院半导体所、复旦大学化学系、浙江大学生物系、厦门大学化学系、解放军防化研究院、西北工业大学等，大都还处在实验室阶段，可喜的是，由王平教授负责的浙江大学科研人员经过十几年研制成功的“电子鼻”肺癌诊断仪于 2006 年 12 月底通过了浙江省科技成果鉴定[9]。研究成果达到国际领先水平。测试者只需对着“电子鼻”呼一口气，把气体吹入“电子鼻”的气袋中，仪器就能基于对多种肺癌诊断的标志性气体，在 30 分钟内查出是否患有肺癌。经过在浙大医学院附属邵逸夫医院的 30 例临床测试，判断成功率近 80%[10]。

1.4 本文的主要工作和章节安排

1.4.1 本文的主要工作

本文通过对传感器节点在硬件平台、软件开发深入研究基础上，完成了传感器节点硬件设计和实现，并根据传感器节点硬件实现了基于微型嵌入式 TinyOS 操作系统的传感器节点软件程序设计，最后完成了基于气敏传感器的无线电子鼻传感器节点实现，并验证了电子鼻节点功能和工作的稳定性。

1.4.2 本文的章节

本论文主要完成无线传感器网络节点的硬件、软件研究，反映到具体研究工作上就是物理上实现节点、节点软件设计和实现基于气敏传感器的无线电子鼻传感器节点。论文研究的主要内容包括：

① 微型嵌入式 TinyOS 操作系统。微型嵌入式 TinyOS 操作系统是面向无线传感器网络的新型操作系统,在软件体系结构上体现了轻量线程技术、主动消息通信技术、事件驱动模式、组件化编程等，这些技术有助于提高无线传感器网络的性能，发挥硬件的特点，降低其功耗，简化应用程序的开发。对它进行研究是实现良好节点程序设计的前提。

② 无线传感器网络节点硬件设计和实现。对基于 MSP430、ATmega128L 处理器芯片和 CC2420 无线收发芯片进行节点设计方案分别进行了研究，实现了节点处理器模块、无线通信模块设计；并对传感器节点能量供应模块以及 Sink 节点串口接入接口进行了设计，最终完成了传感器节点的硬件设计。

③ 无线电子鼻传感器网络节点程序设计。传感器节点通信是节点间组成网络的基础，多跳通信是实现节点数据传输的关键。通过对 nesC 语言、传感器节点消

息数据结构的研究，实现基本的节点通信，并进一步实现节点多跳通信。

④ 完成针对气体浓度检测的无线电子鼻传感器节点的实现，通过实验验证节点功能和工作的稳定性。

1.5 本章小结

本章对无线传感器网络基础理论、关键技术、节点及 802.15.4 协议国内外发展现状和电子鼻技术研究现状做了简述，引入无线电子鼻传感器网络节点的研究意义，简述了文章主要工作和章节安排。

2 微型嵌入式 TinyOS 操作系统

2.1 TinyOS 操作系统简介

TinyOS 是一个开源的嵌入式操作系统，一种面向无线传感器网络的新型操作系统，它是由加州大学的伯利克分校开发出来的，主要采用 nesC 语言编写。nesC 语言是专门为网络嵌入式系统设计的编程语言，对 C 语言进行了一定的扩展，把组件化、模块化思想和基于事件驱动的执行模式结合起来。通过实现一个包含事件驱动执行、弹性并发型和面向组件程序执行等特征的编程模式，来满足这个领域的程序设计的特定要求。TinyOS 操作系统在软件体系结构上体现了轻量线程技术、主动消息通信技术、事件驱动模式、组件化编程等，这些技术有助于提高无线传感器网络的性能，发挥硬件的特点，降低其功能，简化应用的开发[13]。

TinyOS 是基于一种组件（Component-Based）的架构方式，使得能够快速实现各种应用。TinyOS 的程序采用的是模块化设计，所以它的程序核心往往都很小（一般来说核心代码和数据大概在 400 Bytes 左右），能够突破传感器存储资源少的限制，这能够让 TinyOS 很有效的运行在无线传感器网络上并去执行相应的管理工作等。另外，TinyOS 本身提供了一系列的组件，可以很简单方便的编制程序，用来获取和处理传感器的数据并通过无线电来传输信息。可以把 TinyOS 看成是一个可以与传感器进行交互的 API 接口，它们之间可以进行各种通讯。

TinyOS 在构建无线传感器网络时，它会有一个基地控制台，主要是用来控制各个传感器子节点，并聚集和处理它们所采集到的信息。TinyOS 只要在控制台发出管理信息，然后由各个节点通过无线网络互相传递，最后达到协同一致的目的，比较方便。

2.2 TinyOS 组件模型

TinyOS 包含了经过特殊设计的组件模型，其目标是高效率的模块化和易于构造组件型应用软件。对于嵌入式系统来说，为了提高可靠性而又不牺牲性能，建立高效的组件模型是必需的。组件模型允许应用程序开发人员方便快捷地将独立的组件组合到各层配置文件，并在面对应用程序的顶层(top-level)配置文件中完成应用的整体配置。

nesC 作为一种 C 语言的组件化扩展，可表达组件以及组件之间的事件命令接口。在 nesC 中，多个命令和事件可以成组地定义在接口中，接口则简化组件之间的相互连接。在 TinyOS 中，每个模块有一组命令和事件组成，这些命令事件成为该模块的接口。换句话说，一个完整的系统说明书就是一个其所要包含的组件列

表加上对该组件间相互关联的说明。TinyOS 的组件有四个相互关联的部分：一组命令处理程序句柄、一组事件处理程序句柄、一个经过封装的私有数据帧，一组简单任务。任务、命令和事件处理程序在帧的上下文中执行并切换帧的状态。为了易于实现模块化，每个组件还声明了自己使用的接口及其要用信号通知的事件，这些声明将用于组件的相互连接。图 2.1 所示为一个支持多跳无线通信的组件集合与这些组件之间的关系。上层组件对下层组件发命令，下层组件对上层组件发信号通知事件的发生，最低层的组件直接和硬件打交道。

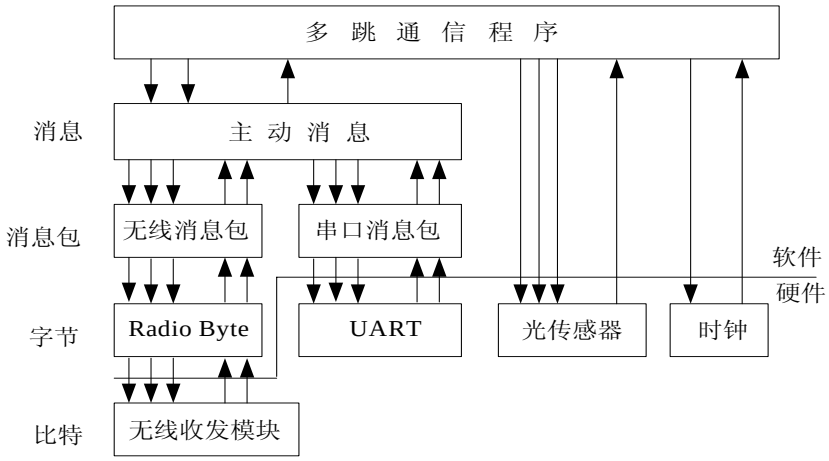

图 2.1 多跳无线通信的组件集合与组件之间的关系

Fig.2.1 The relation between multi-hop wireless communication component congregation and components

TinyOS 采用静态分配存储帧，这样在编译时就可以决定全部应用程序需要的存储器空间。帧是一种特殊的复合 C 语法的结构体，它不仅采用静态分配而且只能由其所属的组件直接访问。TinyOS 不提供动态的储存保护，组件之间的变量越权访问检查是在编译过程中完成的。除了允许计算存储器空间要求的最大值，帧的预分配还可以消除与动态分配相关的额外开销，并且可以避免与指针相关的错误。另外预分配还可以节省执行时间的开销，因为变量的位置在编译时就确定了，而不是通过指针动态地访问其状态[4]。

在 TinyOS 中，命令是对下层组件的非阻塞请求。典型情况下，命令将请求的参数储存在本地的帧中，并为后期的执行有条件地产生一个任务。命令也可以调用下层组件的命令，但是不必等待长时间的或延迟时间不确定的动作发生。命令必须通过返回值为其调用者提供反馈信息，如缓存区溢出返回失败等。

事件处理程序被激活后，就可以直接或间接地处理硬件事件。这里首先要对程序执行逻辑的层次进行定义。越接近硬件处理的程序逻辑，则其程序逻辑的层

次越低，处于整个软件体系的下层。越接近应用程序的程序逻辑，则其程序逻辑的层次越高，处于整个软件体系的上层。命令和事件都是为了完成在其组件状态上下文中出现的规模小且开销固定的工作。最底层的组件可以直接处理硬件中断的处理程序，这些硬件中断可能是外部中断、定时器事件或者计算器时间事件。事件的处理程序可以存储信息到其所在的帧，可以创建任务，可以向上层发送事件发生的信号，也可以调用下层命令。硬件事件可以出发一连串的处理，其执行方向，既可以通过事件向上执行，也可以通过命令向下调用。为了避免命令/事件链的死循环，不可以通过信号机制向上调用命令。

任务是完成 TinyOS 应用主要工作的轻量级线程。任务具有原子性，一旦运行就要运行至完成，不能被其他任务中断。但任务的执行可以被硬件中断产生的事件中断。任务可以调用下层命令，可以向上层发信号通知事件发生，也可以在组件内部调度其他任务。任务执行的原子特性，简化了 TinyOS 的调度设计，使得 TinyOS 仅仅需要分配一个任务堆栈就可以保存任务执行中的临时数据。该堆栈仅由当前执行的任务占有。这样的设计对于存储空间受限的系统是高效的。任务在每个组件中模拟了并发性，因为任务相对于事件而言是异步执行的。然而，任务不能阻塞，也不能空等待，否则将会阻止其它组件的运行。

2.2.1 TinyOS 组件类型

TinyOS 中的组件通常可以分为以下三类：硬件抽象组件、合成组件、高层次软件组件。硬件抽象组件将物理硬件映射到 TinyOS 组件模型。RFM 射频组件(如图 3.1 所示)是这种组件的代表，它提供命令以操纵与 RFM 收发器相连的各个单独的 I/O 引脚，并且发信号给事件将数据位的发送和接收通知其他组件。该组件的帧包含射频模块当前的状态，如收发器处于发送模式还是接收模式、当前的数据传输率等。RFM 处理硬件中断并根据操作模式将其转化为接收(RX)位事件或发送(TX)位事件。在 RFM 组件中没有任务，这是因为硬件自身提供了并发控制。该硬件资源抽象模型涵盖的范围从非常简单的资源(如 I/O 引脚)到十分复杂的资源(如加密加速器)。

合成硬件组件模拟高级硬件行为。如 Radio Byte 组件（如图 2.1）。它将数据以字节为单位与上层组件交互，以位为单位与下面的 RFM 模块交互。组件内部的任务完成数据的简单编码或者解码工作。从概念上讲，该模块是一个能够直接构成增强型硬件的状态机。从更高层次上看，该组件提供了一个硬件抽象模块，将无线接口映射到 UART 设备接口上。提供了与 UART 接口相同的命令，发送信号通知相同的事件，处理相同的数据，并且在组件内部执行类似的任务(查找起始位或符号、执行简单编码等)。高层次软件模块完成控制、路由以及数据传输等。

2.2.2 硬件/软件边界

TinyOS 的组件模型使硬件/软件边界能够比较方便地迁移，因为 TinyOS 所采用的基于事件的软件模型是对底层硬件的有效扩展和补充。另外，在 TinyOS 设计中采用的固定数据结构大小、存储空间的预分配等技术都有利于硬件化这些软件组件。从软件迁移到硬件对于无线传感器网络来说是特别重要的，因为在无线传感器网络中，系统的设计者为了满足各种需求，需要获得集成度、电源管理和系统成本之间的折中方案。

2.2.3 TinyOS 组件示例

TinyOS 中一个典型的组件有四个相互关联的部分：内部帧、命令、事件处理程序句柄、命令和用于消息处理组件的任务大多数组件，都会提供用于初始化和电源管理的命令，另外，它还提供了初始化一次消息传输的命令，并且在一次传输完成或一条消息达到时，向相关组件发送消息。任务、命令和事件处理程序在帧的上下文中执行并切换帧的状态。为了易于实现模块化，每个组件都声明自己使用的接口及要信号通知的事件，这些声明将用于组件的相互连接[4][13]。

2.2.4 TinyOS 组件组合

在 TinyOS 中，组件在编译时被连接在一起，消除了不必要的运行期间的系统开销。为了便于组合，在每个组件文件的开始描述该组件的外部接口。在这些文件中，组件实现了要提供给外部的命令和要处理的事件，同时也列出了要发信号通知的事件及其所使用的命令。从逻辑上讲，可把每个组件的输入输出看成 I/O 引脚，就好像组件是一种物理硬件。组件的向上和向下的接口的这种完整描述被编译器用于自动生成组件的头文件。

2.3 TinyOS 通信模型

2.3.1 主动消息概述

主动消息模式 Active Messages (AM)是一个面向消息通信的高性能通信模式，早期一般应用于并行和分布式计算机系统中。在主动消息通信方式中，每一个消息都维护一个应用层的处理器。当目标节点收到这个消息后，就会把消息中的数据作为参数，并传递给应用层的处理器处理。应用层处理器一般完成消息数据的解包操作、计算机处理或发送影响消息等工作[4][14][15]。在这种情况下，网络就像是一条包含最小消息缓冲区的流水线，从而消除了一般通信协议中经常的缓冲区处理方面的困难情况。为了避免网络拥塞，还需要消息处理能够实现一步执行机制。

尽管主动消息起源于并行和分布计算领域，但其基本思想适合无线传感器网络的需求。主动消息的轻量体系结构在设计上同时考虑了通信框架的可扩展性和

有效性。主动消息不但可以让应用程序开发者避免使用忙等方式等待消息数据的到来，而且可以在通信与计算之间形成重叠，这可以极大地提高 CPU 的使用效率，并减少节点的能耗。

2.3.2 主动消息的设计与实现

在无线传感器网络中采用主动消息机制的主要目的是使无线传感器网络节点的计算和通信重叠，让软件层的通信原语与无线传感器网络节点的硬件能力匹配，充分节省无线传感器网络节点的有限存储空间。可以把主动消息通讯模型看作一个分布式事件模型，在这个模型中各个节点相互间可并发地发送消息。

为了让主动消息更适合于无线传感器网络的需求，要求主动消息至少提供三个最基本的通信机制：带确认信息的消息传递，有明确的消息地址，消息分发。应用程序可以进一步增加其他通信机制以满足特定的要求。如果把主动消息通信实现为一个 TinyOS 的系统组件，则可以屏蔽下层的通信硬件，为上层应用提供基本的、一致的通信原语，方便应用程序开发人员开发各种应用。

在基本通信原语的支持下，开发人员可以实现各种功能的高层通信组件，如可靠性传输的组件、加密传输的组件等。这样上层应用可以根据具体需求，选择合适的通信组件。在无线传感器网络中，由于应用的千差万别和硬件的功能有限，TinyOS 不可能提供功能丰富的通信组件，而只能提供最基本的通信组件，最后由应用程序选择或定制所需要的特殊通信组件。

由于 TinyOS 不支持动态内存分配，所以在主动消息通信组件中保存了一个固定尺寸且预先分配好的缓存队列。如果一个应用程序需要同时存储多个消息，则需要在其私有数据帧上静态分配额为的空间以保存消息。事实上在 TinyOS 中，所有的数据分配都是在编译时确定的。

2.3.3 主动通信的缓存管理机制

在 TinyOS 的主动通信实现中，如何实现消息的存储管理对通信效率的显著影响。 当数据通过网络到达节点时，首先要进行缓存，然后主动消息的分发(dispatch)层把缓存中的消息交给上层应用处理。在许多情况下，应用程序需要保留缓存中的数据，以便实现多跳(multi-hop)通信。

如果节点上的系统不支持动态内存分配，则实现动态申请消息缓存就比较困难。TinyOS 为了解决该问题，要求每个应用程序在消息被释放后，能够返回一块未用的消息缓存，用于接收下一个即将到来的消息。在 TinyOS 中，各个应用程序之间的执行是不能互相抢占的，所以不会出现多个未使用的消息缓存发生冲突，这样TinyOS主动消息通信组件只需要维持一个额外的消息缓存用于接收下一个消息。

2.3.4 主动消息的显式确认消息机制

由于 TinyOS 只提供 best-effort 消息传递机制，所以在接收方提供确认反馈信息给发送方以确定发送是否成功是很重要的。采用简单的确认反馈机制可极大简化路由和可靠传递算法。

在 TinyOS 中，每次消息发送后，接收方都会发送一个同步的确认消息。在 TinyOS 主动消息层的最底层生成消息确认包，这样比在应用层生成确认消息包省开销，反馈时间短。为了近一步节省开销，TinyOS 仅仅发送一个特殊的立即数作为确认消息的内容。这样发送方可以在很短的时间内确定接收方是否需要重新发送消息。从总体上看，这种简单的显式确认通信机制适合无线传感器网络的有效资源，是一种有效的通信手段。

2.4 TinyOS 任务事件驱动与并发模型

事件驱动机制就是事件直接或间接地由硬件中断产生，TinyOS 接收到事件后，立即执行此事件对应的事件处理函数。事件处理可以抢占当前运行的任务，应用于时间要求严格的应用中，但是产生事件的中断源极其有限，无法满足多任务的实时应用[16][17]。

TinyOS采用任务和事件驱动, 相结合的两级并发模型，任务机制任务由用户应用程序定义，可以由应用程序或事件处理程序创建。任务由task关键字定义，具体定义语法为：task void NeedTask(){…}。任务由post关键字创建，具体语法为：post NeedTask()。创建任务时，TinyOS的调度器将任务加入任务队列的队尾。核心调度策略中的任务调度器把此任务加入任务队列后就立即返回，任务则延迟执行。在等待执行的任务队列中，各个任务之间采用FIFO原则进行调度，任务间不能相互抢占。任务机制没有实时调度能力，适用于非抢占、时间要求不严格的应用。TinyOS任务事件驱动并发模型如图2.2所示。

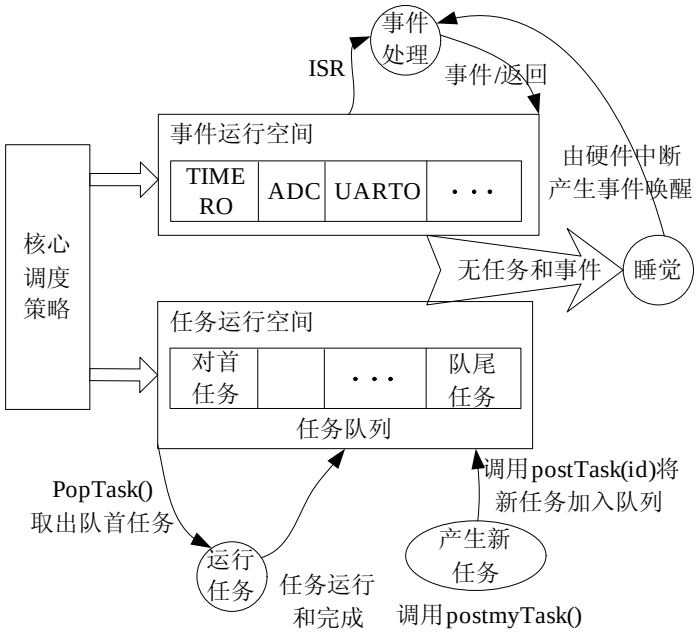

图 2.2 TinyOS 任务事件驱动并发模型示意图

Fig.2.2 The sketch of TinyOS tasks with event-driven model

2.5 TinyOS 内核调度机制与策略分析

TinyOS 是专为无线传感器网络设计的轻量级、低功耗的嵌入式操作系统。它采用基于组件的架构方式，以快速实现各种应用。TinyOS 的编程语言为 nesC，其程序采用模块化设计，这使得它能适应硬件的多样性，允许应用程序重用一般的软件服务与抽象。目前，它已经被成功的应用到多种硬件平台上，具有很高的应用价值和研究意义。

2.5.1 TinyOS 的调度机制

在无线传感器网络中，单个节点的硬件资源有限，如果采用传统的进程调度方式，首先硬件无法提供足够的支持；其次，由于节点的并发操作比较频繁，而且并发操作执行流程又很短，这也使得传统的进程/线程调度无法适应。

TinyOS 采用比一般线程更为简单的轻量级线程技术和两层调度方式：高优先级的硬件事件句柄(hardware event handlers)以及使用 FIFO 调度的低优先级的轻量级线程((task，即 TinyOS 中的任务)。任务一般用于对时间要求不高的应用中，它实际上是一种延迟计算机制。任务之间互相平等，没有优先级之分，所以任务的调度采用简单的 FIFO。任务间互不抢占，而事件(大多数情况下是中断)可抢占。即任务一旦运行，就必须执行至结束，当任务主动放弃 CPU 使用权时才能运行下一个任务，所以 TinyOS 实际上是一种不可剥夺型内核。内核主要负责管理各个任

务，并决定何时执行哪个任务[27]。

为了减少中断服务程序的运行时间，降低中断响应延迟，中断服务程序的设计应尽可能精简，以此来缩短中断响应时间。TinyOS 把一些不需要在中断服务程序中立即执行的代码以函数的形式封装成任务，在中断服务程序中将任务函数地址放入任务队列，退出中断服务程序后由内核调度执行。内核使用一个循环队列来维护任务列表。默认情况下，任务列表大小为 8。图 2.3 是任务队列为空时的情形，图 2.4 表示有三个任务在队列中等待处理。

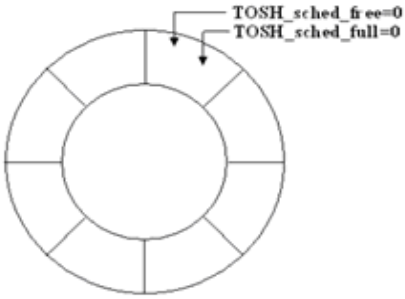

图 2.3 任务队列为空

Fig.2.3 The task queue is empty

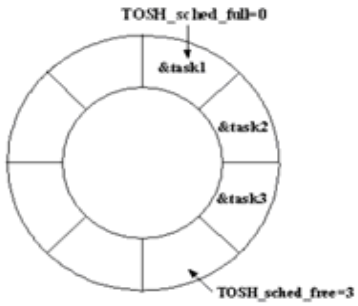

图 2.4 任务在队列中等待处理

Fig.2.4 The task in the queue waiting to be handled

内核根据任务进入队列的先后顺序依次调度执行，即调度算法为简单的 FIFO。TOSH_run_next_task()函数负责从队列中取出指针 TOSH_sched_full 所指的任务并执行。 内核在一个无限循环中调用 TOSH_run_next_task()，当队列不为空时依次执行所有任务函数。

由前所述，TinyOS 调度模型有以下特点：

① 任务单线程运行到结束，只分配单个任务栈，这对内存受限的系统很有利。

② 没有进程管理的概念，对任务按简单的 FIFO 队列进行调度。对资源采取预先分配，且目前这个队列里最多只能有 7 个待运行的任务。

③ FIFO 的任务调度策略是电源敏感的。当任务队列为空时，处理器休眠，随后由外部事件唤醒 CPU 进行任务调度。

④ 两级的调度结构可以实现优先执行少量同事件相关的处理，同时打断长时间运行的任务。

⑤ 基于事件的调度策略，只需少量空间就可获得并发性，并允许独立的组件共享单个执行上下文。同事件相关的任务集合可以很快被处理，不允许阻塞，具有高度并发性。

⑥ 任务之间互相平等，没有优先级的概念。

2.5.2 TinyOS 的调度机制不足

传感器网络中，节点典型的三个任务为：接收待转发的路由数据包、将接收到的数据包转发出去、处理本地的传感数据并将其发送出去。节点上任务的多少取决于节点处理数据的方式。如果节点只是直接把原始数据发往基站，则任务大多数是通信路由任务；如果节点在本地采集数据并处理后才往基站发送，则本地处理任务比较多。当节点上待处理的任务超过其处理能力时，就会发生过载。对于前一种情况，如果节点上发送数据的频率过高或者网络节点密度过大导致通信任务过多时，就可能发生过载；对于后者，如果本地待处理的数据量过大或者本地任务发生频率过高，也会导致过载的发生。

另外，当节点上中断发生频率很高，导致 CPU 除了进行中断处理外不执行其它任何任务时也会出现过载(也叫接收活锁)。当系统处理任务的速率低于任务发生的频率时，任务队列(当前只能存放 7 个任务)很快就满了，则会导致任务的丢失。对于本地的传感采集速率，我们可以人为调节控制，例如降低节点采样频率；但对于通信路由任务的发生，则不太好人为干涉。这时如果发生过载，则直接导致通信数据包吞吐量的下降[11]。

2.6 本章小结

本章着重研究微型嵌入式 TinyOS 操作系统，深入了解了 TinyOS 组件模型、通信模型、事件驱动机制和并发模型以及 TinyOS 内核调度机制以及不足，更好的理解 TinyOS 是进行传感器节点程序设计的基础。

3 无线传感器网络节点硬件设计和实现

无线传感器网络节点一般由处理器模块、无线收发模块、传感器模块和能量供应模块四分部构成，在设计上一般要求微型化、可扩展性、稳定安全、成本低。传感器节点的应用背景决定节点设计中所使用的传感器的种类、精度和采样频率，同时也对无线通信使用的频段、传输距离、数据收发速率提出要求[13]。

3.1 处理器模块设计

处理器模块是无线传感器网络节点的计算核心，所有的设备控制、任务调度、能量计算和功能协调、通信协议、数据整合和数据转储程序都在这个模块的支持下完成。

3.1.1 处理器选型

无线传感器网络节点的处理器一般要求外形尽量小、集成度尽量高、功耗低而且支持睡眠模式、运行速度应尽量快、要有足够的外部通用 I/O 端口和通信接口、成本要尽量低以及有安全性保证。

目前使用较多的有 TI 公司的 MSP430 超低功耗系列处理器、ATMEL 公司的 AVR 系列单片机，它们不仅功能完整、集成度高，而且根据存储容量的多少提供多种引脚兼容，使开发者很容易根据应用对象平滑升级系统。另外作为 32 位嵌入式处理器 ARM 处理器，功耗低、速度快，集成度也相当高，而且地址空间非常大，可扩展大容量的存储器[17]。

3.1.2 基于 MSP430f149 处理器模块电路设计

MSP430 系列微控制器是美国德州仪器（Texas Instruments）公司推出的功能强大的超低功耗 Flash 型 16 位 RISC 指令集微处理器。MSP430 系列单片机能在 8MHz 晶体的驱动下，实现 125ns 的指令周期。MSP430 具有非常高的集成度，单片集成了多通道 12bit 的 A/D 转换、片内精密比较器、多个具有 PWM 功能的定时器、斜边 A/D 转换、片内 USART、看门狗定时器、片内数控振荡器（DCO）、大量的 I/O 端口以及大容量的片内存储器，单片可以满足绝大多数的应用需要。MSP430f149 采用“冯-诺依曼”结构，具有一个硬件乘法器、6 个 I/O 端口、1 个精确的模拟比较器、8 路 12 位 A/D 转换器、片内看门狗定时器、2 个串行通信接口和 60KB 的 FlashROM，2KB RAM。MSP430f149 还具有强大的扩展功能，其具有 48 个 I/O 引脚，每个 I/O 口分别对应输入、输出、功能选择、中断等多个寄存器，使得功能口和通用 I/0 口可以复用，另外价格低廉，所以适合于做传感器节点的处理器[[19]。

针对 MSP430f149 这款芯片设计的节点电路原理图如图 3.1 所示。

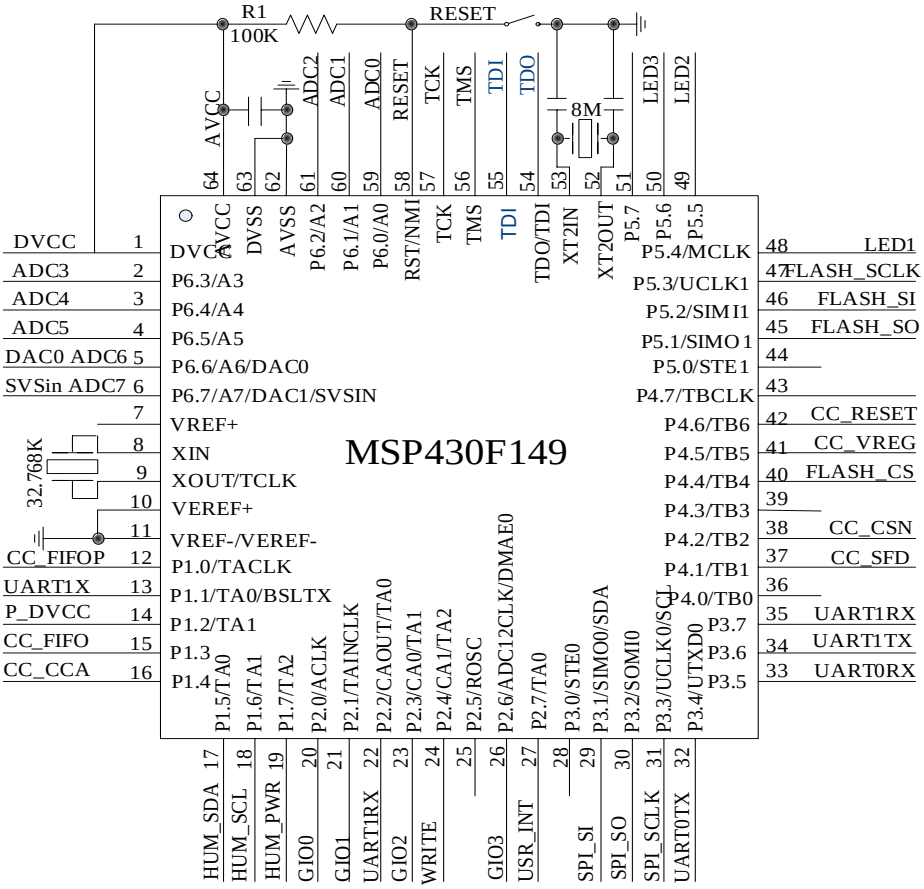

图 3.1 MSP430 节点处理器和外部 IO 接口原理图

Fig.3.1 MSP430 processor node and external IO interface schematic

系统基本的工作原理如下：MSP430F149 及其内部集成的 12 位 AD 负责对数据进行采集和处理，负责控制整个系统工作。如果传感器节点为感知节点那么它将通过湿度、温度传感器或光电二极管等传感器采集有用数据，然后通过处理器控制无线收发模块把信息发送出去，若为中转节点，通过无线收发模块实现数据的转发，若节点为 Sink 节点，处理器控制串口实现数据到网关或 PC 机的传输[21]。在节点设计中使用的 I/O 接口表 3.1。

表 3.1 基于 MSP430F149 节点设计使用的 I/O 接口列表

Table .3.1 MSP430F149 node design I/O interface list

MSP430 的外部引脚及其在原理图中 的网络标号	在节点中 I/O 功能定义
P5.4~P5.6（LED1~LED3）	系统工作指示灯
P5.3(FLASH_SCLK) P5.2(FLASH_SI) P5.1(FLASH_SO) P4.4(FLASH_CS)	连接外部的串行 Flash，串行 Flash 是通过同步串行接口（SPI）进行控制。详细控制方式参考具体芯片资料。
P4.6(CC_RESET) P4.5(CC_VREG) P4.4(CC_CS) P4.2(CC_CSN) P4.1(CC_SFD) P3.3(SPI_SCLK) P3.2(SPI_SO) P3.1(SPI_SI) P1.4(CC_CCA) P1.3(CC_FIFO) P1.0(CC_FIFOP)	CC_RESET 引脚用于芯片的复位，CC_VREG 用于使能 CC2420 稳压，其余八个引脚用于处理器控制 CC2420 工作实现数据收发，其中 SPI_SCLK、SPI_SO、SPI_SI 三个引脚用于 SPI 控制，其中处理器工作为主机模式，CC2420 工作在从机方式，CC_CCA 用于检查无线信号是否空闲，CC_SFD 用于数据收发的帧开始符，CC_FIFOP、CC_FIFO 用于数字的输入输出，它们的完成会产生中断事件。
TCK、TMS、TDI、TDO	作为 JTAG 引脚使用
P1.7(HUN_PWR) P1.6(HUM_SCL) P1.5(HUM_SDA)	用于读取控制湿度传感器 HUMIDITY/ TEMP SENSOR
ADC0~ADC7	模数转换器外部引脚工作方式参考芯片资料
P2.0(GIO0)、P2.1(GIO1) P2.3(GIO2)、P2.3(GIO3)	提供外部上拉电阻高电平
P2.4(SEMSOR_ID)	连接外部的一个 64 位 ID 芯片（8 位厂商号，48 位有效数据，8 位 CRC 校验位）。
RESET	复位键
UART1RX、UART1TX	用于扩展工作
UART0RX、UART0TX	一方面用于调试及数据收集接口；另一方面用于 Flash 的编程工作
DVCC	电源

3.1.3 基于 ATmega128L 处理器模块电路设计

Atmel 公司的 Atmega128L 处理器采用增强的 RISC 的处理核心，指令集更丰富，执行速度更快。具有 128KBFlash，够编程 10000 次，同时具有 4KB 的 SRAM 和 KB 的 ERPROM、53 个通用 I/O 口线、32 个通用工作寄存器、实时时钟 RTC、4 个灵活的具有比较模式和 PWM 功能的定时器/ 计数器(T/C)、两个 USART、面向字节的两线接口 TWI、8 通道 10 位 ADC(具有可选的可编程增益)、具有片内振

荡器的可编程看门狗定时器、SPI 串行端口、与 IEEE1149.1 规范兼容的 JTAG 测试接口(此接口同时还可以用于片上调试)，以及六种可以通过软件选择的省电模式。空闲模式时 CPU 停止工作，而 SRAM、T/C、SPI 端口以及中断系统继续工作；掉电模式时晶体振荡器停止振荡，所有功能除了中断和硬件复位之外都停止工作，寄存器的内容则一直保持；省电模式时，异步定时器继续运行，以允许用户维持时间基准，器件的其他部分则处于睡眠状态；ADC 噪声抑制模式时，CPU 和所有的 I/O 模块停止运行，而异步定时器和 ADC 继续工作，以减少 ADC 转换时的开关噪声；Standby 模式时，振荡器工作而其他部分睡眠,使得器件只消耗极少的电流，同时具有快速启动能力；扩展 Standby 模式则允许振荡器和异步定时器继续工作。器件是以 Atmel 的高密度非易失性内存技术生产的。片内 ISP Flash 可以通过 SPI 接口、通用编程器或引导程序多次编程[20]。

针对 Atmega128L 这款芯片设计的节点电路原理图如图 3.2 所示。

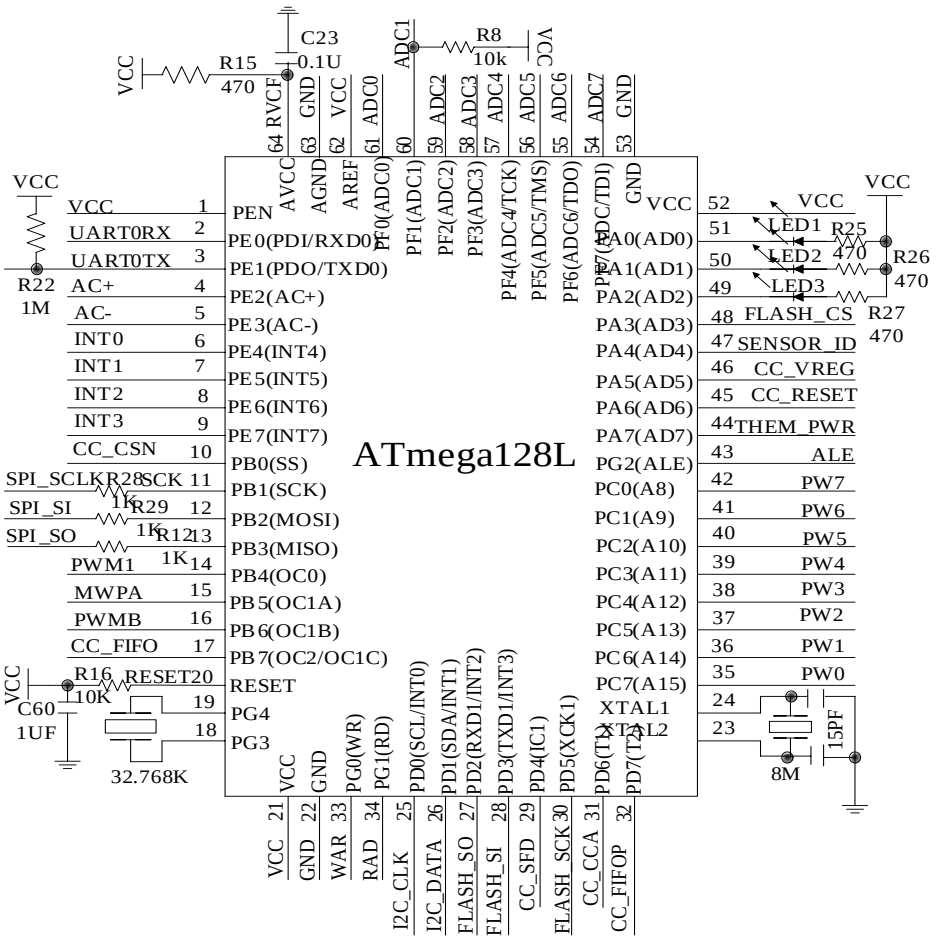

图 3.2 Atmega128L 节点处理器和外部 IO 接口原理图

Fig.3.2 Atmega128L processor node and external IO interface schematic

系统基本工作工作原理：Atmega128L 处理器模块，负责控制整个系统工作，控制无线通信模块实现感知数据的收发，其内部集成的 12 位 AD 负责对数据进行采集和处理。如果节点为感知节点那么它将通过湿度、温度传感器或光电二极管等传感器采集有用数据，然后通过处理器控制无线收发模块把信息发送出去，若为中转节点，通过无线收发模块实现数据的转发，节点作为 Sink 节点，处理器控制串口实现数据到网关的传输。通过使用开关量控制传感器板上传感器的通断，保证只有在需要采集数据时传感器处于工作状态。处理器模块可以工作在各种省电模式，达到能量的节省。

表 3.2 基于 ATmeaga128L 节点设计使用的 IO 接口列表

Table.3.2 ATmeaga128L node design IO interface list

ATmega128L 外部引脚及其在原理图中的网络标号	在节点中 I/O 功能定义
PA0~PA2（LED1~LED3）	系统工作指示灯
PD5（FLASH_SCLK） PD3（FLASH_SI) PD2（FLASH_SO) PA3（FLASH_CS）	连接外部的串行 Flash，串行 Flash 是通过同步串行接口（SPI）进行控制。详细控制方式参考具体芯片资料。
PA6（CC_RESET) PA7（CC_VREG） PB0（CC_CSn） PD4（CC_SFD) PB1（SPI_SCLK） PB3（SPI_SO）、PB2（SPI_SI） PD6（C_CCA） PB7（CC_FIFO） PE6（CC_FIFOP)	CC_RESET 引脚用于芯片的复位，CC_VREG 用于使能 CC2420 稳压，其余八个引脚用于处理器控制 CC2420 工作实现数据收发，其中 SPI_SCLK、SPI_SO、SPI_SI 三个引脚用于 SPI 控制，其中处理器工作为主机模式，CC2420 工作在从机方式，CC_CCA 用于检查无线信号是否空闲，CC_SFD 用于数据收发的帧开始符，CC_FIFOP、CC_FIFO 用于数字的输入输出，它们的完成会产生中断事件。
TCK、TMS、TDI、TDO	作为 JTAG 引脚使用
PC0~PC7(PW0~PW7)	传感器供电/断电开关
PF0~PF7（ADC0~ADC7）	模数转换器外部引脚工作方式参考芯片资料
PE4~PE7（INT0~INT3）	外部中断引脚
PA4(SENSOR_ID)	连接外部的一个 64 位 ID 芯片
RESET	复位键
PD0~PD1	I2c 总线数据传输
UART0RX、UART0TX	一方面用于调试及数据收发接口； 另一方面可用于 Flash 的编程工作编成数据的输出输入
DVCC	电源

3.2 无线通信模块设计

3.2.1 CC2420 芯片

CC2420 是 IEEE802.15.4 标准的低成本、低功耗单片高集成度的解决方案。它工作在 ISM 免费频带上，工作频率为 2.4GHz。CC2420 符合欧洲 ETSI EN 300 328、EN 300 440 class 2，美国 FCC CFR47 15 部分标准和日本 ARIB STD-T66 标准。CC2420 的主要特点：具有 2MChips/s 直接扩频序列基带调制解调和 250kbps 的有效数据速率；适合简化功能装置和全功能装置操作；低电流消耗：接收 19.7mA，发射 17.4mA；低电源电压要求：使用内部电压调节器时 2.1-3.6V，使用外部电压调节器时 1.6-2.0V；可编程输出功率；独立的 128 字节发射、接收数据缓冲器；电池电量可监控；QLP-48 封装,外形尺寸只有 7×7mm。可以看出：该芯片具有良好的性能，尤其是它极低的电流消耗和封装尺寸，可以满足无线传感网络中节点体积小、功耗小、成本低等特点，具有广泛的应用前景[22]。

3.2.2 CC2420 与处理器模块的硬件接口

CC2420 为信息包处理提供广泛的硬件支持，数据缓冲器、发射、数据加密、数据证明、空闲信道评估、链路质量指示和信息包实时资料等，这些特点减少了主控制器的工作量，使 CC2420 可与低成本微处理器相接。CC2420 的四线串行 SPI 接口引脚功能如表 3.3 所示，它是设计单片机电路的依据，充分发挥这些功能是设计无线通信产品的前提。

表 3.3 CC2420 硬件接口引脚

Table.3.3 CC2420 interface pins

引脚	功能
FIFOP	数字输出
FIFO	数字输入、输出
CCA	空心道估计
SFD	帧开始分隔符
SI	SPI 数据输入
SO	SPI 数据输出
SCLK	SPI 时钟
VREG_EN	稳压使能
RESETn	复位
CSn	SPI 使能

一个典型的 CC2420 应用电路主要是将该射频芯片与微处理器连接起来，配以

晶振和负载电容、输入/输出匹配元件和电源电压、动耦电容等很少的外部元件。CC2420 通过简单的四线(SI、SO、SCLK、CSn) 与 SPI 兼容串行接口配置，这时 CC2420 是受控的。ATmega128、MSP430 的 SPI 接口工作在主机模式，它是 SPI 数据传输的控制方，CC2420 设为从机工作方式。CC2420 与 ATmega128L、MSP430 处理器的电路接口引脚 如表 3.4 示。

表 3.4 CC2420 与处理器接口引脚使用

Table.3.4 The use of pins between CC2420 and processor

ATmega128L 引脚	CC2420 引脚	MSP430f149
PA6	RESETn	P4.6
PA7	VREG_EN	P4.5
PB0	CSn	P4.2
PB1	SCLK	P3.3
PB2	SI	P3.1
PB3	SO	P3.2
PB7	FIFO	P2.0
PD4	SFD	P4.1
PD6	CCA	P2.1
PE6	FIFOP	P1.0

基于 CC2420 无线收发芯片的典型应用电路如图 3.3，其接口适合与 ATmeaga128L、Msp430F149 处理器。

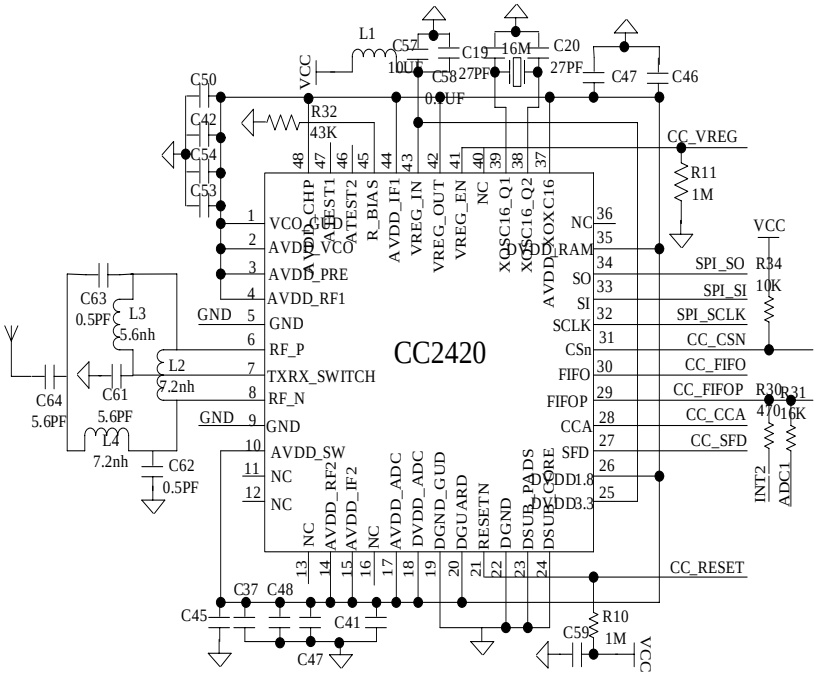

图 3.3 基于 CC2420 无线收发芯片的典型应用电路

Fig.3.3 The CC2420 radio transceiver chip typical application circuit

3.3 能量供应模块设计

节点供电一般使用能量有限的电池供电，在实验室中，需要调试和测试可以通过供电板供电工作。如图 3.4。

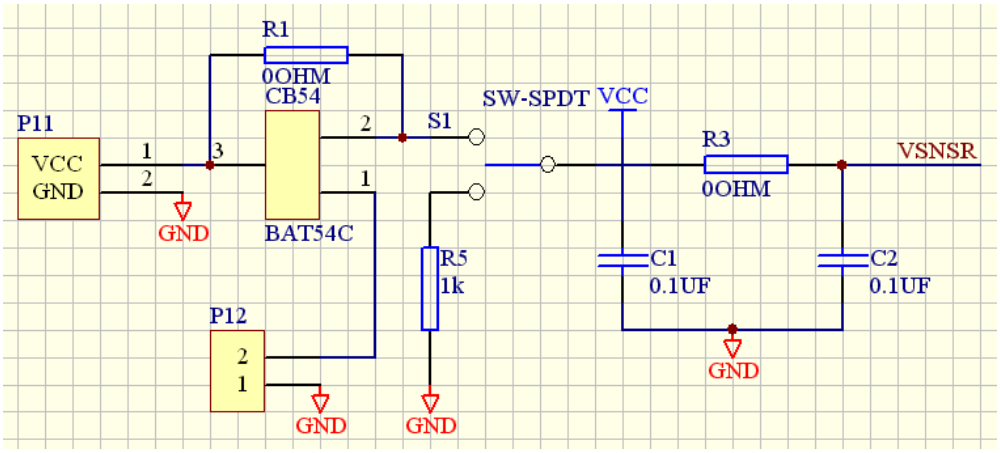

图 3.4 电源供应电路

Fig.3.4 Power supply circuit

传感器电路板电源供应一般需要某个传感器采集数据的时候，才对其进行供电，这就需要采用关断/打开方式实现电源控制，具体实现[21]如图 3.5。

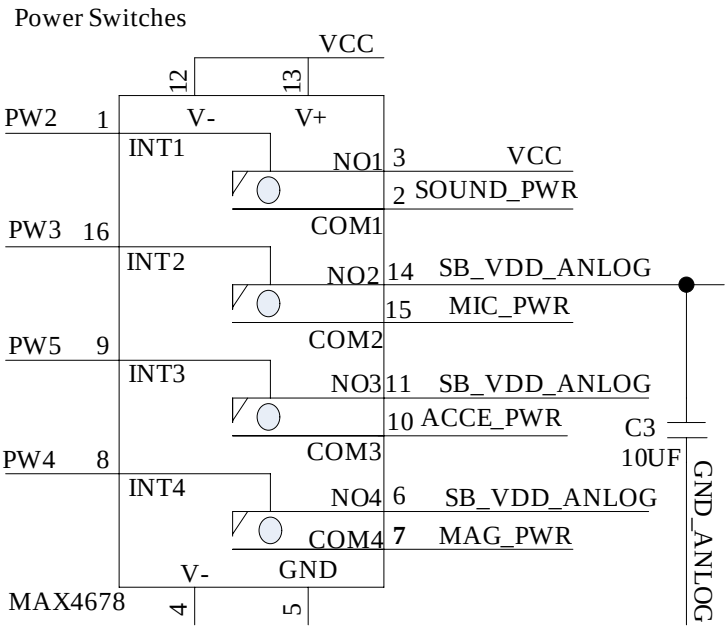

图 3.5 传感器板电源开关电路

Fig.3.5 Sendor board power switch circuit

3.4 Sink 节点串口接口设计

3.4.1 RS232 接口设计

Sink 节点通过串口把数据传输给用户计算机上，单片机处理器与 PC 机的串口连接一般使用 RS232 接口，其典型应用电路如图 2.6 RS232 接口电路。

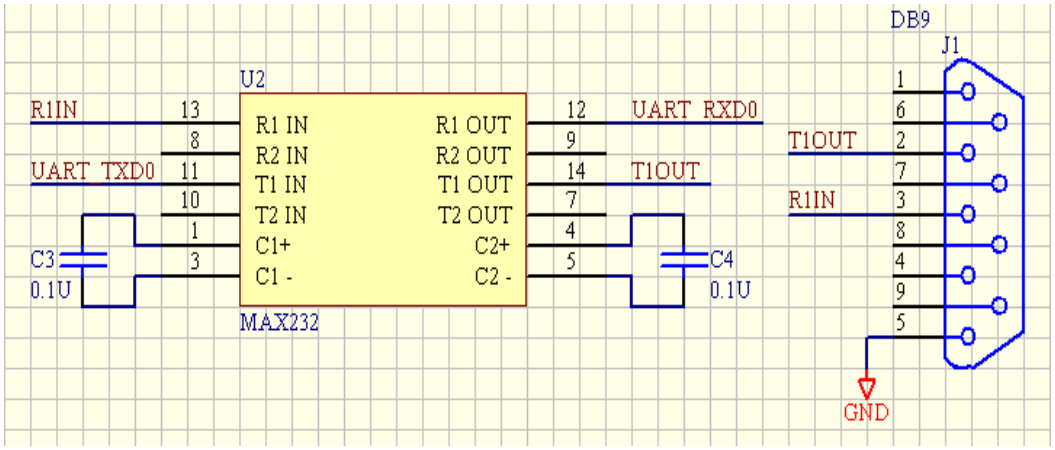

图 3.6 RS232 接口电路

Fig.3.6 RS232 interface circuit

3.4.2 USB 接口设计

由于处理器中使用串口发送数据，所谓的USB接口是指接到PC机上通过USB接口实现，为此需要串口转USB接口芯片把处理器串口成USB口，目前使用广泛的就是FT2232C芯片[23]。其中FT2232C可以提供两个串口转换，因此可以使用其中一个串口进行是数据传输，另一串口用于数据下载，这就需要使用控制引脚控制允许哪个串口工作。典型电路主要模块如图3.7。

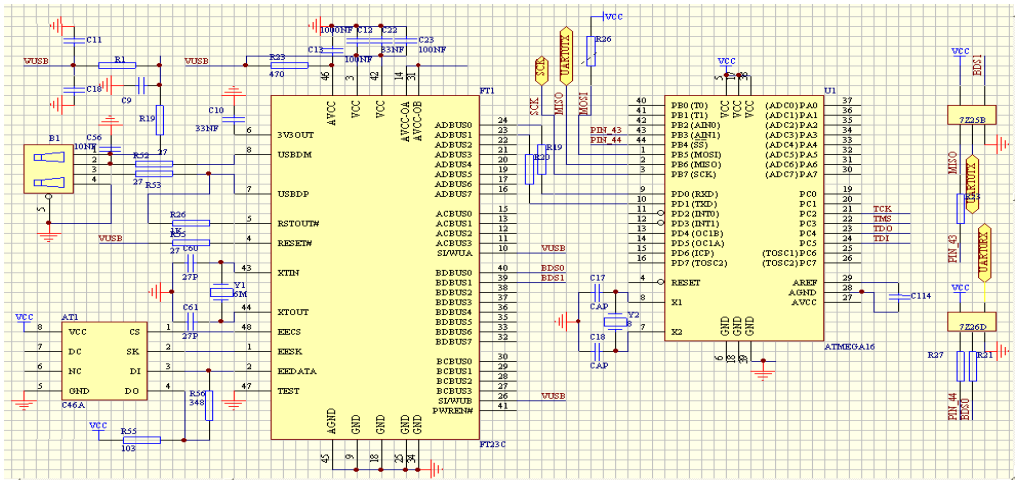

图 3.7 串口转 USB 接口电路

Fig.3.7 UART to USB interface circuit

3.5 其他外围电路

由于传感器节点实现的功能相比处理器的一般应用要复杂的多，所以一般需要多扩存储单元，此处传感器节点设计使用 flash AT45041 芯片实现。另外对于无

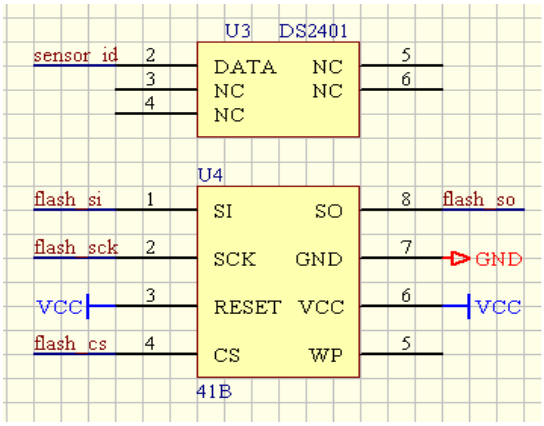

图 3.8 Flash 存储器扩展与序列号生成电路

Fig.3.8 Flash memory expansion and serial number generation circuit

线传感器网络，通信要不断的进行，存在大量的数据包在节点之间发送，为了保证数据包的唯一性，一般把数据赋予唯一的序列号，一般使用 DS2401 芯片实现。电路原理图如图 3.8 所示。

3.6 节点电路板设计

根据上面所述，本文采用 ATmega128L＋CC2420 方案进行节点 PCB 设计，所完成的 PCB 图如图 3.9.

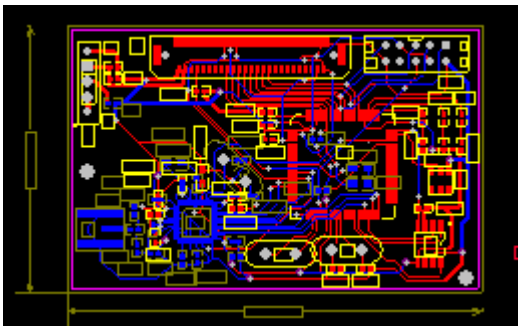

图 3.9 节点 PCB

Fig.3.9 Node PCB

3.7 本章小结

本章主要研究了有关无线传感器网络节点的硬件设计。其中简述了基于 MSP430F149、ATmega128L 处理器模块设计原理、基于 CC2420 无线收发模块设计原理、电源模块以及 Sink 节点串口接入电路，并使用 ATmega128L＋CC2420 方案进行节点的 PCB 制作，实现物理节点。

4 无线电子鼻传感器网络节点程序设计

4.1 nesC 语言

nesC 语言定义了一个支持事件驱动系统的组件模型：这个模型提供的双向接口简化了事件流，支持弹性的软硬件接口，并且有效地避免了虚函数和动态组件构造；定义了一个简单但是表现力很强的并发模型，还有可扩充的编译器分析：nesC 的编译器会在编译期间检测绝大多数数据竞争。这个组合使得应用程序可以在资源有限的情况下实现高度并发操作；另外，在提高可靠性、控制代码大小的精确程序分析和建立实际应用程序的表达能力之间，nesC 提供了一个独一无二的平衡点，除了静态数据竞争检测之外，nesC 的编译器还执行静态组件实例化操作，整体程序内联，还有减少死代码。函数指针和动态内存分配等特征支持复杂的应用程序和牢固的用户通信，但是它们会阻碍静态分析，所以我们在 nesC 中禁止采用这些特征[13]。

一个 nesC 语言编写的应用主要包括三部分：基于 C 语言的声明和定义，接口类型，组件。基于 nesC 语言应用程序由许多功能独立且相互联系的软件组件构成。一个组件一般会提供一些接口，接口可以看作是组件实现的一组函数声明。接口既可以是命令和事件也可以是单独定义的一组命令事件。基于 nesC 语言编写的程序框架如图 4.1。

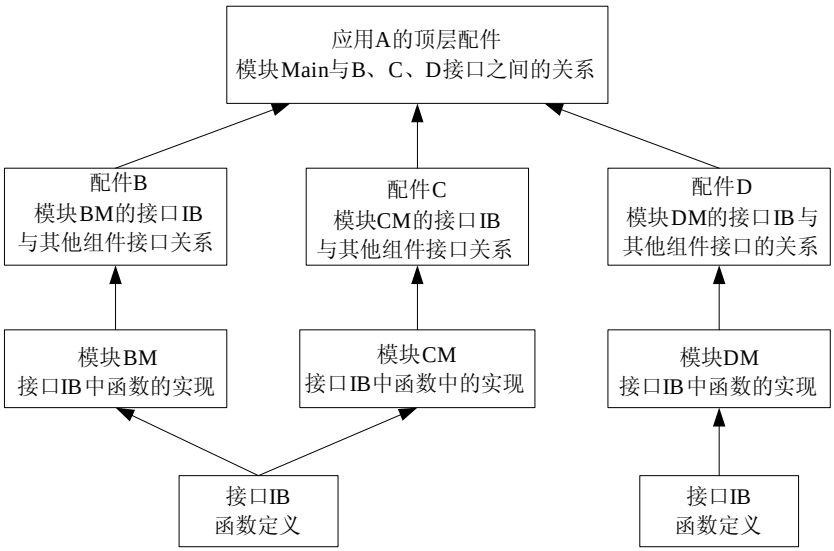

图 4.1 基于 nesC 语言应用程序框架

Fig. 4.1 Application Framework Based on nesC Language

无线传感器网络应用程序和硬件结合得非常紧密，而且每个节点一次只能运行一个应用程序。这一点导致无线网络传感器应用程序设计具备如下三个特点：静态地了解所需的资源需求：不采用通用操作系统；应用程序由一组可重用的系统组件和专门的应用程序代码组成；软硬件的界线不确定；随应用程序和硬件平台而变化。

nesC 必须解决以下一些问题：节点是用作数据采集和现场环境控制的，由传感器跟环境的交互来驱动，和我们传统的通用计算机不一样，其用途决定了在设计过程中要注意以下两点：

第一，节点不是由和人的交互驱动的，也不是由批处理驱动的，基本上是由事件驱动的，并且会根据环境的变化做出相应的反应。

第二，事件的到达和数据的处理是并发活动，这需要设计一个恰当方法来进行并发控制，这种并发控制很容易异致竞争情况等潜在错误。有限的资源，由于受到小尺寸、低造价和低功耗等限制，节点只有很有限的物理资源。

4.2 电子鼻传感器节点程序框架流程

电子鼻传感器节点采集完数据后，产生中断事件，引发无线收发模块发送电子鼻采集到的数据，中继传感器节点，转发数据，即采集到的数据通过传感器节点多跳通信方式，最后把数据传送到 Sink 节点，最终 Sink 节点通过串口发送到网关或 PC 机上。在 PC 机完成采集数据的处理。功能实现过程如图 4.2。

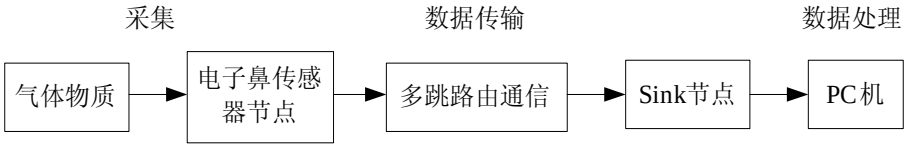

图 4.2 电子鼻功能实现过程

Fig.4.2 Realization of E-nose

传感器节点上的软件负责完成现场数据的采集以及通过无线通信模块将采集数据包通过无线方式传送。节点遵循睡眠-被唤醒-正常工作的工作模式。在睡眠状态下，处理器停止工作，而 SRAM、SPI 端口以及中断系统继续工作，无线模块处于低电流的接收状态。当无线通信模块接收到 Sink 节点或是临近节点发出的命令后，节点被唤醒，处理器对命令进行节点号判断，如果命令的对象是当前节点，则节点进入工作状态，否则节点对命令进行转发后再次进入睡眠状态。

传感器节点经过初始化、建链后直接进入休眠模式。当主节点收到中断请求时触发中断，激活节点，发送或接收信息包，处理完毕后继续进入休眠状态，等

待有请求时再次激活。若有多个从节点同时向主节点发送请求，主节点来不及响应处理而丢掉一些请求，则从节点在发现自己的请求没有得到响应后几秒钟再次发出请求直到得到主节点的响应为止。传感器节点采用串行口通信模式，程序中采用中断的方法来完成数据的接收和发送。

传感器节点主程序处理流程如图 4.2 所示

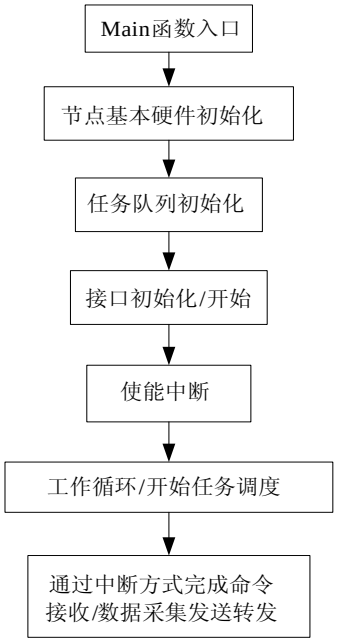

图 4.3 节点程序流程

Fig .4.3 Node program procedure

4.3 TinyOS 程序 main 函数入口

在 TinyOS 中进行程序开发，其函数入口从根本上讲仍为 main 函数。但是此时的 main 被 RealMain 函数所代替，这种方法类似 Windows 程序开发中使用 WinMain 函数做为 Window 程序开发的函数入口。在 RealMain 函数函数要完成一些的初始化、中断设置、任务的调度运行等，简述代码如下：

```
{
    int main() __attribute__ ((C, spontaneous)) {
    call hardwareInit();
    call Pot.init(10);
    TOSH_sched_init();
    call StdControl.init();
    call StdControl.start();
```

```
        __nesc_enable_interrupt();
        while(1) {
        TOSH_run_task();
    }
```

由第一段代码可知真正的函数入口仍为 main 函数，上面代码类似与 VC 开发程序中使用 MFC 框架，这是 TinyOS 为每一个开发者都准备好了的代码。而 main 函数的代码如下：

```
configuration Main {
            uses interface StdControl;
                        }
  implementation
{
    components RealMain, PotC, HPLInit;
    StdControl = RealMain.StdControl;
    RealMain.hardwareInit -> HPLInit;
    RealMain.Pot -> PotC;
 }
```

虽然上述 main 函数代码为配置文件，但是说明了 main 函数的 stdControl 接口使用的 RealMain 函数的 stdControl 接口。恰恰是通过这种方式，我们编写自己的程序代码，然后通过编写一个配置文件与 RealMain 函数关联起来，使 main 函数最终调用的接口转化成调用我们自己编写的组件中的接口，最终引入我们自己程序代码的运行。

4.4 TinyOS 数据包解析

串口数据包就是节点采集后的数据最终要通过串口发送到 PC 机上的原始数据包，其中包含感知数据和一些信息包说明信息。TinyOS 的数据包最大长度可以拥有 255 字节，原始数据包的两端都是以 0x7E 打包的。这用于检测数据流的开始与结束。原始数据包使用了一种逃避字节 0x7d，这是在一个字节有效载荷数据和保留字节码相同时的情况下需要，如帧同步字节 0x7e。在这种情况下，需要在有效数据前加上逃避字节，而后原同步字节与 0x20 进行异或。例如，一有效载荷数据字节 0x7e 会出现在数据包作为 0x7d 0x5e。对 XP 的机器，在数据流中多字节值是字节对调即为小端模式。例如，2 字节 UART 的地址栏（0x007e），将显示为 7E00 中字节流。

4.4.1 原始数据包

以下图和表描述了原始数据包。

表 4.1 原始数据包说明

Table 4.1 Raw data packet explanation

同步字节	包类型	有效数据	同步字节
0	1	2 ……n-1	n

字节#	区 域	详 细 描 述
0	数据包帧同步字节	一直为 0X7E
1	包类型	有下面几种已知的数据类型： • P_PACKET_NO_ACK (0x42)-用户无需 ACK 信号。 • P_PACKET_ACK (0x41)-用户需要 ACK。包括前缀字节。接收机必须发送一份P_ACK反应前缀字节作为内容。 • P_ACK (0x40)–ACK 信号反应 P_PACKET_ACK。包括前缀字节作为其内容。 • P_UNKNOWN (0xFF)–未知包类型。
2 – n-1	有效数据	在大多数情况下将是一个 tinyos 信息但是长度不一，它的描述如下面。
n	同步字节	一直为 0X7E

4.4.2 TinyOS 信息数据包

有效载荷数据类型是典型一类 TinyOS 消息，所确定的结构为 TOS_Msg。这种数据结构的定义如下：

```
typedef struct TOS_Msg
{
  uint16_t addr;
  uint8_t type;
  uint8_t group;
  uint8_t length;
  int8_t data[TOSH_DATA_LENGTH];
  uint16_t crc;                    //以上区域用于在无线中收发
```

```
    uint16_t strength;              //以下区域不在无线中收发，而用于内部计算
    uint8_t ack;
    uint16_t time;
    uint8_t sendSecurityMode;
    uint8_t receiveSecurityMode;
  } TOS_Msg;
```

关于 TOS_Msg 数据包详细描述如下：

表 4.2 TinyOS 信息包说明

Table 4.2 TinyOS message packet explanation

地 址		消息类型	组 ID	数据长度	数 据	数据校验	
0	1	2	3	4	5……-n-2	n-1	n

字节#	区 域	详 细 描 述
0-1	消息地址	其中三种可能的值类型： •广播地址（0xffff）信息发送给全部节点。 •UART 的地址（0x007e）-消息，从节点到网关的串行端口。所有来袭信息将产生这个地址。 •节点地址-几点接收消息的唯一地址。
2	消息类型	每种应用都有其相应的消息类型： • AMTYPE_XUART = 0x00； • AMTYPE_MHOP_DEBUG = 0x03； • AMTYPE_SURGE_MSG = 0x11； • AMTYPE_XSENSOR = 0x32； • AMTYPE_XMULTIHOP = 0x33； • AMTYPE_MHOP_MSG = 0xFA；
3	组 ID	节点组成的组都有一个唯一的 ID，默认情况下为 125 (0x7d)。只有组 ID 相同的节点可以相互通信。
4	数据长度	记录有效数据的长度，不包括 CRC 校验和帧同步字节。
5-n-2	有效数据	实际的信息内容。
n-1-n	校验码	两个字节代码以确保消息的完整性。

4.4.3 传感器节点多跳信息包

TinyOS 消息包内的有效数据为嵌入某节点一具体应用的原始数据包。在大多数的情况下，尤其是在动态自组网的应用中需要使用的多跳消息协议。关于多跳消息的定义如下。多跳信息格式如下面的表格。

```
typedef struct MultihopMsg {
    uint16_t sourceaddr;
    uint16_t originaddr;
    int16_t seqno;
    uint8_t hopcount;
    uint8_t data[(TOSH_DATA_LENGTH - 7)];
} __attribute__ ((packed)) TOS_MHopMsg;
```

表 4.3 多跳信息包说明

Table.4.3 multi-hop message packet explanation

Source Address		Origin Address		Sequence Number		HopCount	App data
0	1	2	2	4	5	6	7-n

字节#	区 域	详 细 描 述
0,1	源地址	转发节点的地址
1,2	源地址	原始数据包节点地址
4,5	序列号	用于确定路由，并计算丢失包
6	条数	用于确定路由，节点走过数据
7-n	应用数据	应用数据。特定应用数据。如 Tgs_Msg（见下文）。

4.4.4 电子鼻消息数据包

多跳消息内部的应用数据就是相对于用户实际应用的原始数据。消息的类型取决于 TinyOS 消息中区域中的第 2 个字节。电子鼻的消息类型为 AMTYPE_SURGE_MSG (0x11).消息格式定义如下：

```
typedef struct Tgs_Msg {
    uint8_t type;
    uint16_t reading;
    uint16_t parentaddr;
    uint32_t seq_no;
    uint8_t Vref;
    uint8_t tgs1;
    uint8_t tgs2;
    uint8_t tgs3;
```

```
    uint8_t tgs4;
    uint8_t tgs5;
    uint8_t tgs6;
} __attribute__ ((packed)) Tgs_Msg;
```

有关 Tgs_Msg 消息详细描述如下：

表 4.4 电子鼻信息包说明

Table 4.4 E-node message packet explanation

Type	Reading		father Addr		Sequence Number				Vref	Tgs1	Tgs2	Tgs3	Tgs3	Tgs5
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

字节#	区 域	详 细 描 述
0	类型	类型的不同一位行动的不同，已知的类型有： • SURGE_TYPE_SENSORREADING (0x00)； • SURGE_TYPE_ROOTBEACON (0x01)； • SURGE_TYPE_SETRATE (0x02)； • SURGE_TYPE_SLEEP (0x03)； • SURGE_TYPE_WAKEUP (0x04)； • SURGE_TYPE_FOCUS (0x05)； • SURGE_TYPE_UNFOCUS (0x06)；
1，2	读	没有使用
3，4	父地址	父亲节点的地址
5-8	序列号	前 9 个 bit 位用于表示电压，其余为用于记录最近一次复位所发送的数据包
9	Vref	电压读
10	Tgs1	气敏传感器数据读
11	Tgs2	气敏传感器数据读
12	Tgs3	气敏传感器数据读
13	Tgs4	气敏传感器数据读
14	Tgs5	气敏传感器数据读
15	Tgs6	气敏传感器数据读

4.4.5 采样数据包消息示例分析

综合上面所讲，一个采样电子鼻消息数据包具体如下：

原始数据包：

7E 42 7D 5E 00 11 7D 5D 16 00 00 02 00 9D 00 00 00 00 00 00 00 6F 00 80 DB F9 7D 5E FF FF 01 C8 EC D9 7E

原始数据包我们可以从串口观察到，下面是 TinyOS 消息包，不含逃脱字节：

7E 00 11 7D 16 00 00 02 00 9D 00 00 00 00 00 00 00 6F 00 80 DB F9 7E FF FF 01 C8 EC D9

具体字节详细描述见表 4.5 数据包分析说明：

表 4.5 电子鼻数据包分析说明

Table 4.5 E-nose data packet analysis

字 节	描 述
7E 00	串口地址- 126 (0x00 7E)
22	消息类型-电子鼻消息- 17 (0x11)
7D	组 ID
16	数据长度
00 00	最近转发节点地址 即为节点 0
02 00	最初位节点 2 数据包
9D 00	序列号
00	跳数为 0
00	电子鼻消息类型（00-传感器读）
00 00	没有使用
00 00	父节点地址 00 00
6F 00 80 DB	前 9bit 表示电压 其余为发送数据包数
F9	原始气敏传感器采集数据值
7E	原始气敏传感器采集数据值
FF	原始气敏传感器采集数据值
FF	原始气敏传感器采集数据值
01	原始气敏传感器采集数据值
C8	原始气敏传感器采集数据值
EC D9	校验码

4.5 基于 CC2420 无线通信模块程序设计

传感器节点之间相互通信是实现无线传感器网络组网的基础。传感器节点要正常工作首先要完成无线通信模块的初始化，主要包括处理器接口初始化、内部寄存器的初始化以及其他通信相关参数的设置。

4.5.1 初始化

关于处理器接口的初始化在 TinyOS 中一般通过宏函数来实现：

引脚工作方式（输入、输出）：

```
TOSH_MAKE_MISO_INPUT();      TOSH_MAKE_MOSI_OUTPUT();
TOSH_MAKE_SPI_SCK_OUTP;      TOSH_MAKE_CC_RSTN_OUTPUT();
```

SPI 工作模式（SPI 工作设置）：

```
sbi (SPSR, SPI2X);                    // Double speed spi clock
sbi(SPCR, MSTR);                      // Set master mode
cbi(SPCR, CPOL);                      // Set proper polarity
}
```

上面是通过宏函数实现了接口引脚工作模式方式的设置，在此设置ATmega128L 的 SPI 接口的工作方式，SPI 为主机模式，使用两倍 SPI 时钟，设置 SPI 时钟为 3.6Mhz，使能 SPI 端口。

关于寄存器的设置，CC2420 在复位时对各寄存器均进行了默认的初始化，在使用前还应根据实际应用对相关寄存器的值进行修改。操作中，要执行 SXOSCON 命令选通寄存器开启晶体振荡器，配置 MCTRLO 寄存器为 Ox0AF2 设置自动地址识别、自动确认帧应答、自动 CRC 校验等功能，配置 MDMCTRLl 寄存器为 0x0500 设置关联门限值为 20，并选择发送接收工作模式，配置 IOCFGO 寄存器为 Ox007F 设置 FIFOP 门限至最大值 128，并开启自动识别 PAN ID 标识符功能。

在 CC2420 进行收发通信之前还需对 CC2420 的信道号、PAN 标识符、源地址、目标地址等通信参数进行设置。在 TinyOS 中采用自定义数组的方式实现初始化。下面是在 TinyOS 中完成 TXCTRL 寄存器初始化：

```
gCurrentParameters[CP_TXCTRL] = ((1 << CC2420_TXCTRL_BUFCUR) |
    (1 << CC2420_TXCTRL_TURNARND) | (3 << CC2420_TXCTRL_PACUR) |
    (1 << CC2420_TXCTRL_PADIFF) | (CC2420_TXPOWER
      << CC2420_TXCTRL_PAPWR));
```

4.5.2 CC2420 发送/接收子程序

在初始化完成后，首先判断 CC2420 的 SFD 脚和 FIFOP 脚的状态，当 CC2420 发送接收空闲时，清空 TX_FIFO 发送缓存区并将需要发送的数据包以工 EEE 802.15.4 MAC 层数据帧格式依次写入，此时若检测到 RSSI 有效且 STXONCCA 信道估计显示信道空闲，则写命令选通寄存器 STXONCCA 使能发送功能，发送数据包。

数据包发送时，CC2420 自动在数据包的开始处加上前导码和帧起始分隔符，在数据包末尾加 CRC 检验。由于接收节点的 CC2420 在初始化时 MCTRLO 寄存器设置为自动回复确认帧功能，因此在数据发送完成后，CC2420 将延时等待接收确认帧，若收到目标节点回复的确认帧，则表示发送成功且目标节点接收成功，反之表示目标节点没有收到数据包，重新启动发送数据包。在 TinyOS 中，实现程序所需组件及其调用关系，如图 4.3。

节点需 CC2420 接收数据时，首先写命令选通寄存器 CC2420_ SRXON，使能 CC2420 接收功能。CC2420 接收到数据包时 FIFO 管脚状态发生变化，数据包存入接收 FIFO 缓存区中，接收到一个完整的数据包时 FIFOP 管脚使 ATmega128L 进入

中断。ATmega128L 通过读 RX_FIFO 寄存器依次读出整个数据包，并读取帧长度判断接收到的帧是否有效。接收到的帧包括确认帧、有效数据帧、错误帧，若接收到错误帧则直接丢弃，接收到确认帧和有效数据帧时进行相应的数据处理，当接收到的帧为有效的数据帧且校验正确，则自动回复目标节点数据成功接收的确认帧。

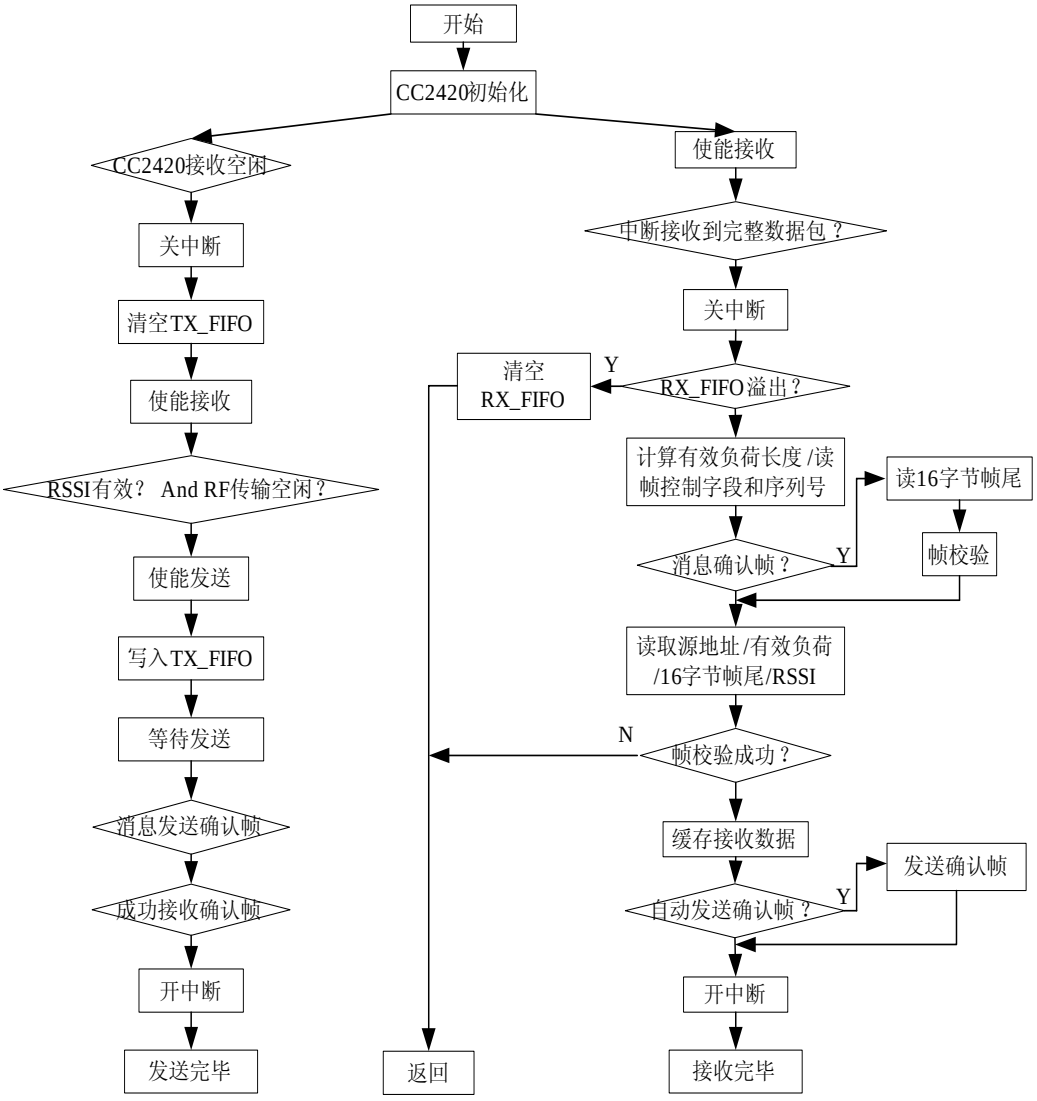

图 4.3 CC2420 发送、接收流程图

Fig .4.3 CC2420 send and receive flow chart

在接收过程中如果 RX_FIFO 寄存器发生下溢或溢出，立即清空 RX_FIF 接收子程序的流程图[24]如图 4.3 所示。在 TinyOS 中，实现程序所需组件及其调用关系，如图 4.4。

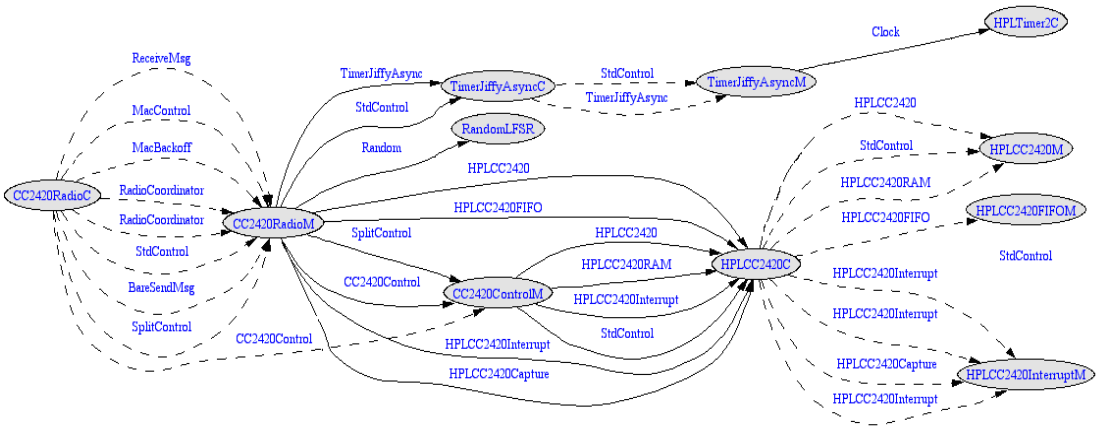

图 4.4 CC2420 组件调用关系

Fig.4.4 The calling relation between CC2420 components

4.6 传感器节点串口通信

串口通信一般只有 Sink 节点使用，主要完成采集数据的发送和查询命令的接收然后通过无线转发给其它节点。

串口的初始化：

```
UART.init() {
outp(0,UBRR0H);
outp(15,UBRR0L);
outp((1<<U2X),UCSR0A);
outp(((1 << UCSZ1) | (1 << UCSZ0)) , UCSR0C);
outp(((1 << RXCIE) | (1 << TXCIE) | (1 << RXEN) | (1 << TXEN))，UCSR0B);}
```

串口发送采集数据：

```
atomic{
      outp(data,UDR0);
      sbi(UCSR0A,TXC);
    }
TOSH_INTERRUPT(SIG_UART0_TRANS) {
      signal UART.putDone(); }
```

串口命令接收：

```
TOSH_SIGNAL(SIG_UART0_RECV) {
      if (inp(UCSR0A) & (1 << RXC))
         signal UART.get(inp(UDR0));
   }
```

4.7 传感器节点 ADC 模块

当节点采集数据，需要通过使用 ADC 通道，实现数据采集，不同的传感器使用不同的 ADC 通道，在 TinyOS 可以使用通道绑定函数来绑定某种传感器使用某一个 ADC 通道。ADC 模块初始化代码如下：

```
atomic {
        outp(((1 << ADIE) | (6 << ADPS0)),ADCSR);
        outp(0，ADMUX);
          }
```

ADC 数据采集：

```
atomic {     outp((TOSH_adc_portmap[port] & 0x1F),ADMUX);    }
sbi(ADCSR,ADEN);
sbi(ADCSR,ADSC);
```

完成 ADC 数据采集、发送完成信号：

```
TOSH_SIGNAL(SIG_ADC) {
     uint16_t data = inw(ADCL)；
     data &= 0x3ff；
     sbi(ADCSR,ADIF);
     cbi(ADCSR,ADEN);
     __nesc_enable_interrupt();
     signal ADC.dataReady(data);
    }
```

4.8 传感器节点多跳通信

现实的无线传感器网络由于节点无线传输距离的限制，绝大多数感知信息都要经过多个节点相互帮助传输即通过多跳的方式来完成，因此实现节点信息多跳通信是实现真正意义上的无线传感器网络的关键。

4.8.1 TinyOS 的 Multi-Hop 组件

TinyOS-1.1 发布了库组件其中包括实现多跳路由组件，用作传感网络的一些具体应用。在 TinyOS 中数据的移动和路径决定引擎分别使用了具有单一接口独立组件，并在它们之间允许其它路径决策，以便将来能够和其它多跳路由更好的整合。程序设计中使用多跳路由协议，很容易移植。因为我们只需针对具体的应用程序使用正确的接口，那么它对我们来时基本上是透明的。 下面是关于多跳库组件的描述。

MultiHopEngineM 和 MultiHopEWMA 是实施多跳路由的两个核心组件，它们通过配置文件 MultiHopRouter 连接在一起。如图为 MultiHopRouter 配置流程图。

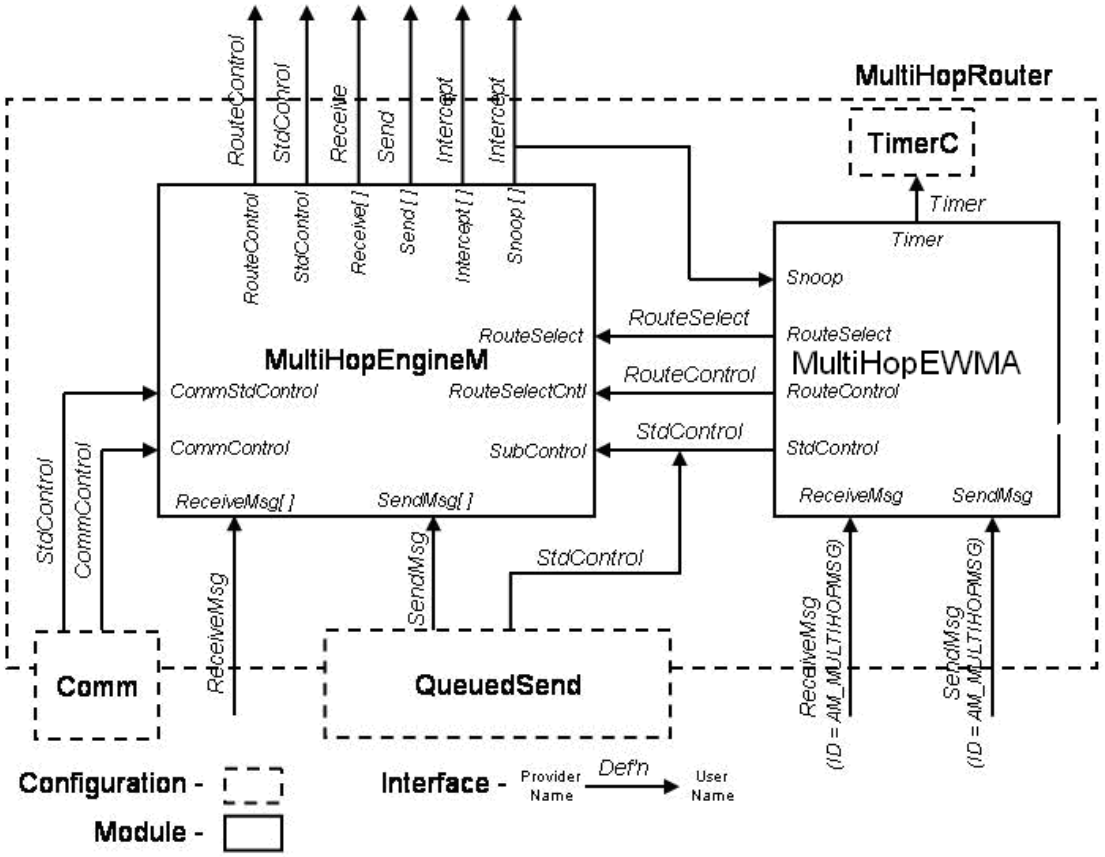

图 4.5 MultiHopRouter 配置流程图

Fig .4.5 The flow chart of MultiHopRouter configure

MultiHopEngineM：提供了所有多跳路由数据包运动逻辑。使用 RouteSelect 接口，它决定被参数化的 SendMsg 端口转发发送下一跳的多跳路径。模块在路由选择上是独立的。只需在算法组件中使用 RouteSelect 和 RouteControl 接口。

MultiHopEWMA：为多跳路由执行提供链接估算及父节点选择（EWMA）机制。该模块通过侦听端口方式监控所有节点收到交通信息，一经收到单跳消息马上更新信息为 AM_MULTIHOPMSG 类型，随后可以将数据包发送给单跳邻域范围内节点。在模块内部管理最近的现有邻居节点和下次到目的地的最短路径。

MultiHopRouter：用于配置连接 MultiHopEngineM 和 MultiHopEWMA 两个组件以及和它们有联系的组件。一般配置接收、传送、中断和侦听端口。其中 MultiHopEngineM 的 SendMsg 接口与 QueuedSend 库组件连接用于排出发送消息；

MultiHopEngineM 的 ReceiveMsg 和 SendMsg 接口通过使用 AM_MULTIHOPMSG 类型的消息，更新单跳路线与邻邦。

组件一般包含下面六种接口：StdControl-标准的控制接口；RouteControl-一种专用与控制监测路由器运作的接口；Reveive[]-最终目的数据包的接收接口；Send[]-本地原始包发送接口；Interrupt[]–用于估计当一个需要被转发的数据包被接收到时转发的交通信息，以及根据返回值，决定具体转发操作；Snoop[]-这个端口可以使用‘中断’接口定义，但有不同的语义。接到转发数据包，但是并不会进行转发。该接口可被动的监听消息的交通状况。

4.8.2 最小跳数的路由协议

基于最小跳数路由协议的基本思想是每个节点记忆自己的一跳邻居内最小跳数比自己小 1 的节点。采用基于最小跳数的路由协议，可以保证任何节点发出的信息都沿着最优的路径向 Sink 节点发送，并且在网络内所引起的信息包数量是最少的。所有节点只要记忆自己的转发节点集和最小跳数就可以实现信息路由。该协议可以在任何规模的网内使用，并且对于移动节点也可以很简单的实现信息转发，另外在节点具有固定发射功率的无线传感器网络中，可以尽最大距离传送，充分利用发射功率[25][26][27]。

基于最小跳数的无线传感器网络路由协议的实现主要包括两个阶段：最小跳数场的建立过程阶段和数据采集阶段。

① 最小跳数场的建立过程

在传感器网络中每一个节点都有一个唯一标识的 ID，并且 Sink 节点一般初始化为 0，其余节点跳数为一极大值。在监测区域布置好网络节点后，Sink 节点向外以 flooding 方式广播建立最小跳数场的消息包 A，这个消息包主要包含消息的类型、节点的网络 ID 和节点的当前跳数。A 消息被临近节点接收到这些节点将自己的跳数设置为 1，将 Sink 节点作为自己的转发节点，并返回应答加入信息 Join-A 消息。然后这些所有 1 跳节点发出 2 跳请求消息 B，同样这些消息将被临近节点收到，这些节点将请求消息 B 中包含的最小跳数和自己当前最小跳数比较，如果两者相等，则将该信息包的发送节点加入到自己的转发节点集，如果小于自己当前的最小跳数，则将最小跳数设置为该信息包的最小跳数，并将发送节点消息加入转发节点集，同时向 Sink 节点返回加入消息 Join-B，否则丢弃该包。这样的过程一直持续下去，最后网络中每一个节点都获得了自己到 Sink 节点的最小跳数和转发节点集，同时 Sink 节点根据各节点返回的加入消息，建立了一个包含网络中所有节点及其转发节点集的网络拓扑图。

通过此种方式建立通信链路并不能说明对应的两个节点能够进行对等通信，也不能说明该临时通信链路有比较高的通信质量，为此协议还应使用质量评估的

方法，测量通信链路的数据发送功率，从而达到对通信链路的通信质量的评估。在评估过程中，具有较高数据发送功率的临时链路被保留下来，成为固定链路。

② 数据采集阶段

数据下发过程：当 Sink 节点有命令需要向目标节点发送时，Sink 节点需要搜索一条从 Sink 节点到达目标节点的路径，它首先根据网络拓扑图查找到目标节点，根据目标节点的转发节点集，为其选择一个上一跳节点，然后再为该上一跳节点选择它的上一跳节点，这样逐跳向上查找，直到找到 Sink 节点，所有选出的节点就组成一条从 Sink 节点到达目标节点的路径，Sink 节点就可以通过这条路径将命令下发到目标节点。

数据上传过程：如果网络中某一节点有数据需要上传到 Sink 节点时，它将在转发节点集中选择一个节点进行转发，如果该节点不能到达，则选择转发节点集中另一节点进行转发，转发节点接到数据后，按同样过程将数据包转发给上一跳节点，数据包依次向上传送，最终到达 Sink 节点。

无论是数据上传还是数据下发过程，都有为某一节点选择上一跳节点的问题，如果一个节点总是选择同一个上一跳节点，则所有经过该节点的数据都通过同一个上一跳节点进行转发，这样造成该上一跳节点能量消耗过快。为了避免上述情况，可以在选择上一跳节点时，采用随机选择的方式，在转发节点集中随机选择一个节点作为上一跳节点，这样转发节点集中的每个节点被选中的机会都是均等的，增加了网络平衡性，避免了局部节点失效，延长了网络生存周期[28][29]。

4.8.3 链路质量估计

单独节点链路质量估计可以通过观察数据包成功和损失的事件来获得。高层次的协议可通过利用单独节点估计建立更好的路由结构。链路质量估计最好是一种对于链路质量上发生潜在大变动能够迅速做出反应，且稳定，内存占用小，便于计算的估计。因为变化反应快速使得层次较高的协议，更好的适应环境的变化和传感器节点数据的流动性[30]。但是，估计必须是稳定的；如果波动剧烈，路由拓扑不可能稳定，一些路由问题如周期、滞留节点，将会很常见。由于存储有限，用于代表邻居节点的足迹记忆估计要尽量小。我们只有通过有限的存储资源，处理能力以及能量来实现链路质量估计[31][32]。

实现链路质量的估计一般使用通过侦听信道的方法[33]。无线通信允许被动的侦听，在传感器网络中，广播的本质也就是执行信道的侦听。侦听失效可以通过每一个数据包中的序列号来判断。侦听估计存在花销，因为某一节点侦听的数据包不一定就是给它的。但是在大多数情况下，数据包可以被允许任何节点收到。各种个样的低功耗侦听存在于文献[36]中，可以在很大程度降低侦听的功耗。还有一种方法就是通过使用接收信号的强度作为链接质量估计的标准[37]。然而由于转

发错误改正和在测量中一些不确定的因素使得通过信号强度来估计链路质量相对较差。

在通过被动侦听信道进行链路质量估计中会发现很微妙的是在数据包成功接收中没有信息损失，更特别的是如果一个节点丢失，它的估计降为 0。可以通过假设每一个节点使用最低限度的数据传输速率，进行数据传输，在许多传感器网络中都可以这样做，而且还可以更好的修正高数据损失率链接。

在文献[37]中可以看到许多链路估计方法，由于节点能量有限、处理能力低、存储容量小在加上环境的影响使能进行估计十分复杂，资源的限制线性回归或卡尔曼滤波不合实际。一般进行估计都是在已经建立好的类似于无线传感器网络实际的环境中，几种简单有效的候选值估计方法一般包括：加权指数移动平均值（EWMA）、时间加权移动平均值、带有指数加权数据包成功/丢失平均值。实现中这些都是参数可进行调整，保证估计的灵活性和稳定性。

4.8.4 邻居节点数据表管理

无论是作为被动监测还是主动探测，一个节点都是通过从它收到的数据包中信息来发现邻居并记录邻居节点信息。链接估算是用来确定哪些节点应考虑在邻居分布式连接图中。控制传输功率，可以调整单元密度，常常是一个特定有效的使用参数，因为它是影响到整个网络的跳计数，延迟信道利用率，网络生存时间和质量的重要环节[39]。

邻居表管理主要由三个部分构成：插入节点，驱逐节点和节点加强。每一个邻居节点都会对消息数据包进行分析，考虑对这个消息源节点进行插入或者加强。如果这个消息源节点存在于邻居表中，将会执行一个加强操作使其保留在邻居表中原来位置。如果邻居表空或者源节点没有存在于邻居表中，那么节点将要决定是丢弃源节点信息还是驱逐表中另外一个节点。在保证稳定的链路质量的前提下，通过算法使邻居表在不管节点密度的情况下保持一种充足邻居节点数量，以获得好的有效的路径。所以邻居表管理要使新的邻居节点加入还要防止弱链接节点污染[36] 。

在算法上我们应专注于邻居节点的发现上。节点周期的侦听数据信息。每一个表的条目中都包括着节点链路估计和路由数据。当一个节点从表中被驱逐，它的链路估计便失效。如果表没有满那么将执行插入操作，然而只有当邻居表满的情况下才执行驱逐操作[41]。

插入策略：当节点收到一个非邻居节点源消息，就必须决定是否插入它。由于它没有在邻居表中，对于它没有任何历史信息可以利用。在某种情况下可以使用数据包地理位置信息或者信号强度，但是地理位置信息经常是缺省的并且没有考虑阻塞的，而信号强度可变性大。可以另寻方法。插入策略应该避免高速率的

插入以保证建立相对稳定的邻居表。通过降低采样率来控制插入速率是一种常见的方法。实现保持一个固定的概率的降低采样速率工作是容易的，但是为每一个节点设定正确的值是相对困难的，因为潜在邻居节点数据可以有很大的变化。一般使用一种自适应降低采样率的方案来设置插入邻居表的采样速率。设置邻居表的大小为 T，邻居节点的数量为 N。我们假设只考虑周期消息是可行的，对于插入所有 N 个邻居产生一个粗率的相等的发现率。对于数据采集，我们可以考虑原始数据包和忽略转发数据包。从平均上说，给每一个节点建立的机会，对于每次接收的 N 个邻居消息最多可以插入 T 个邻居节点到邻居表中。为了对 N 个节点进行估计，在关于估计连续数据流中不同的数值的数据库文献前人已经做了很多优先工作[43]。不过，我们可以只计算某一节点接收消息的平均消息数量。另外，如果节点已经在邻居表中还需要对该节点进行加强。

节点驱逐、加强策略：对节点进行驱逐有几种方法可以选择。其中最简单的就是通过循环方式，不需要实施节点加强操作。鉴于计算机缓存技术，可以选择 FIFO、最近侦听、时钟概率算法。对于 FIFO 算法，节点的驱逐是通过节点进入表的先后次序，不需要节点加强。对于最近侦听算法主要是基于最近侦听到的消息。时钟算法可以进行加强操作，通过设置加强的节点的表中位置标志位，一旦要进行驱逐节点就对邻居表进行扫描，直到发现没有设标的节点出现，进行清除标志位。在进行频率估计算法中，可以有一种简单的方法。维持每一个表记录进入频率数量。一旦节点插入，节点的加强次数便被加 1。如果有节点插入频率次数为 0，那么新的节点可以被插入。否则所有节点次数减 1，新节点被丢弃[42]。

4.8.5 协议执行框架

无线传感器网络的建立过程一般都是通过洪泛方式，网络结构为树状结构，网络的连通性都是通过节点数据包之间的相互通信，例如发送路由请求信息包、路由回答信息包，最终形成复杂树状网络结构 。网络由大量可靠链路和一些不可靠链路以及一些孤立孩子组成。

如图 4.6 展示了通过所有组件实施路由协议过程。

每一个节点维护一个接收（in-bound）链路质量估计。路由建立应立足于发送(out-bound)链接质量估计，这信息将来回馈给邻居节点。作为核心部件的邻居表管理它包含了邻居节点的身份状态和路由入口。表中每一个数据包括 MAC 地址、路由成本、父节点地址、孩子标志、接收链路质量计、发送链路质量计以链路估计数据结构。估计器侦听路由层下面所有在信道上的数据包，邻居表管理器控制着节点的插入、驱逐和加强。路由协议是分布式的以距离矢量为基础通过父节点选择组件实现。父节点选择操作运行并周期性发送路由消息以确定其中一个邻居节点路由信息，也可能采用发送路由消息的方式。路由消息包括父亲地址，到 Sink

节点路由代价估计和邻居节点接待列表。当路由消息被邻居表中的节点接收，相应的入口被更新。不然，邻居表管理器将决定是否对节点进行插入或者放弃更新。数据包根据转发队列中当前的父节点作为目的地址而有选择的被转发。来自采集节点原始数据包经过节点处理，排队等待发送到作为目的节点的父节点。来袭（Incoming）数据包有选择的通过发送队列转发到当前作为目的地址父节点。相应的邻居表入口被标示为孩子节点用来避免父节点选择出现循环。重复转发数据包需要剔除。当检测到转发数据包循环时，则父节点重新选择被触发，以打破数据包循环转发。

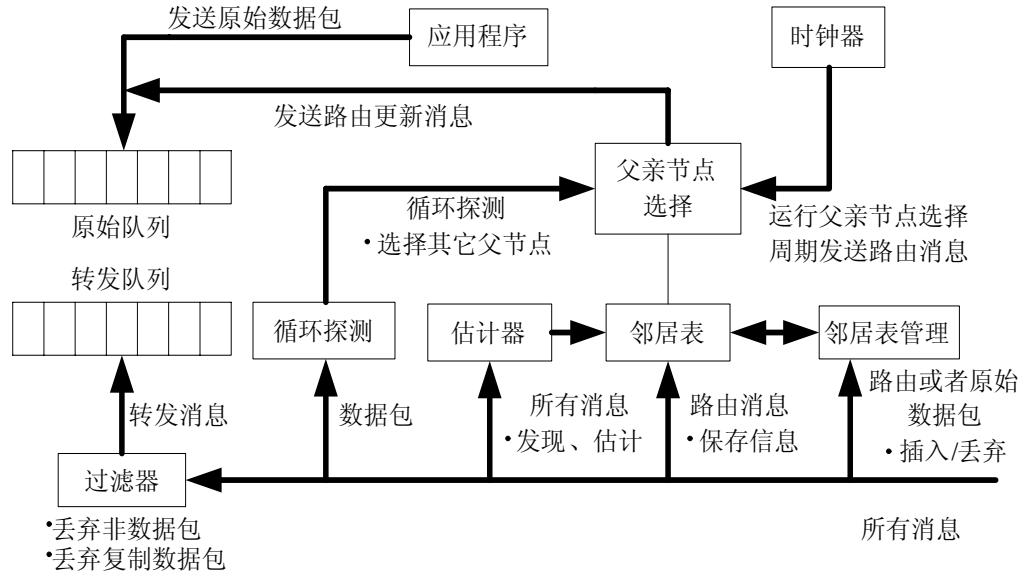

图 4.6 路由组件流程图

Fig .4.6 Message flow chart to illustrate the different components for routing

父节点选择：对于基于距离向量的算法，传感器节点路由损耗可以看作是距离的一种抽象衡量。有可能是到达 Sink 节点的跳数、预期的传输数量等。对于其中某一个节点只有当它的损耗比当前父节点损耗小的时候才能成为潜在的父节点。还有就是只有在链路质量估计低于某一门限的情况下，或者节点信息最终无法到达 Sink 节点，或者父节点选择出现循环，需要重新进行父节点选择。随着时间的推移，到达父节点连接性会不断变差，相应的链路质量也在下降，此时需要进行新的父节点选择。如果节点失去了到父节点的连接，并且没有潜在的父节点，此时，该传感器节点需要声明其没有父亲节点，并声明其路由损耗无穷大。

不管什么路由协议，父节点的变更都会引发路径的改变，要快速适应父节点变更，需要对新路径每一个邻居节点进行估计，因为对于多米诺骨牌效应的路线

改变，当路由的成本非常敏感的时候是可能会影响整个网络的变更。由此为了达到一个更稳定的网络，一般采取周期性变更路由，而不是通过更新通知方式变更，除非探测到循环的出现[39]。

数据包侦听：无线传感器网络是一个广播的媒介，大量信息都可以通过侦听方式进行提取。在路由协议层，每一个节点都是一个路由器。节点通过侦听方式可以获知其孩子信息，这可以防止循环信息；通过侦听相邻节点信息可以很快的获知其父节点信息，这可以防止两跳循环。还可以通过侦听方式裁剪局部网无关的孩子节点，当一个节点收到来自其孩子一不可达到数据包时，它将转发送其孩子的数据包并在其中加上 NO_ROUTE 标志。所有的邻居节点包括其孩子都能侦听到该数据包，并可以获知没有传送路径。事实上，当然也会有相应的反馈信息到树状网络深层内部，以解决计数无穷大问题，重新完成数据的感知。

循环：对于网络中出现数据包转发循环问题，采用简单机制来最大程度上避免成环和当探测到循环打破循环，要比使用冗重复杂的协议算法实现环的内部协调要好[43]。DSDV 路由协议提供了一个很好的在动态网络中避免循环的方法，也可以通过监控孩子节点交通信息和侦听临近孩子节点的中信息包父亲节点地址，确定孩子节点信息以及确定其不能成为潜在父节点，并在邻居表中保存此信息方式，此种方式只需要为邻居表中每一个节点维护这些信息。

脱离网络的节点和网络在进行裁剪用来纾缓陈腐信息的孩子节点会生成无效路径，导致循环产生。循环存在发生的潜在性，并且一定可以被探测到。因为每一个节点本身都是一个路由器和数据源，循环一旦形成，那么节点发送一个数据包并且重新收到该数据包。只要信息队列管理可以避免让转发交通信息没有覆盖原始交通信息，那么循环一旦出现就会被探测到。当探测到循环出现，节点允许抛弃原父节点选择新的父节点或者脱离网络成为孤立节点。

重复数据包剔除：当 ACK 信号丢失的时候，需要数据的重发，此时就可能产生重复数据包。如果不消除重复数据包，可能引起更多的重发或更多其它的内容，还有需要能量消耗。为了避免重复数据包，在路由层原始节点上追加发送者 ID 和在路由头上加上原始序列号。同时，每一个父亲节点在邻居表孩子入口地方保留最近数据包发起者 ID 和原始序列号。

队列管理：在网络中层次较高的节点转发数据远远超过它们自身产生的数据。要小心不能让转发数据占据整个队列，这样会阻止原始数据包的发送和循环回路的探测。我们把消息队列分成两个独立的队列：消息转发队列和原始数据包发送队列。一般情况下，原始数据包发送要少于数据包的转发，可以给发送数据包更高的优先级。

基于 TinyOS 多跳程序设计具体实现中，多跳组件使用及其相互关系，如图 4.6。

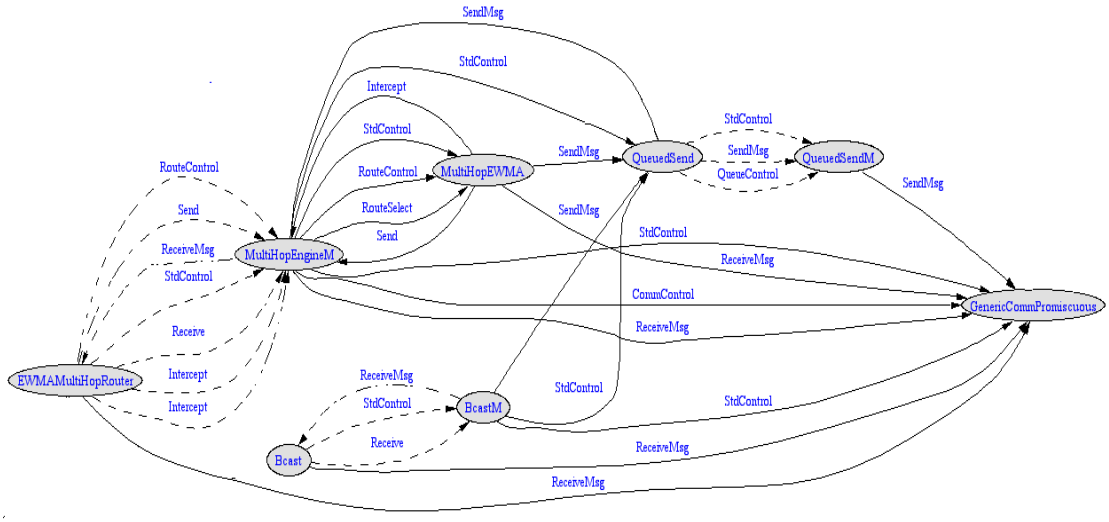

图 4.6 多跳组件使用和组件关系

Fig.4.6 The use of Multi-hop and relation between components

4.9 本章小结

本章节首先介绍了节点程序设计语言 nesC，主要对电子鼻传感器节点程序流程和使用的消息的数据结构进行了详细的分析，进而展开了传感器节点无线通信模块程序设计、串口通信模块、ADC 节点数据采集研究。最后详细分析传感器节点多跳通信的实现。

5 无线电子鼻传感器网络节点实现

电子鼻又称气味扫描仪，是一种智能仿生传感器系统。J. W. Gardner 给电子鼻下的定义是：“电子鼻是由一种有选择性的电化学传感器阵列和适当的识别装置组成的仪器，能识别简单和复杂的气味。”电子鼻通常由一个具有部分专一性的传感器阵列和一个合适的模式识别系统组成，能够定性或定量地检测单一的或混合的气味，还能够用于识别单一成分的气体、蒸汽或其他混合物[44]。

典型的电子鼻系统由气敏传感器阵列、微处理器与接口电路和模式识别系统组成。气敏传感器阵列可以是由分立的气敏传感器构成，也可以是单片集成的气敏传感器阵列。气敏传感器阵列用来感应气体中的化学成分，产生可以用来测量的物理量的变化。

5.1 无线电子鼻传感器模块设计

电子鼻传感器模块主要完成气敏数据的采集，然后通过 ADC 模块进行转换，处理器模块在通过无线方式把数据发送出去。而要使得电子鼻功能更为准确，构成电子鼻的传感器头一般有多个构成气敏传感器阵列。电子鼻传感器模块主要结构如图 5.2。

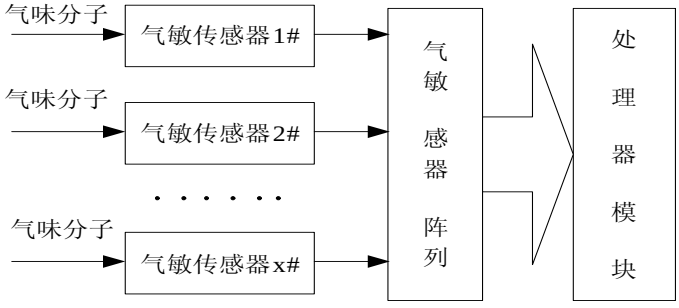

图 5.1 传感器模块结构图

Fig.5.1 Sensor module structure

电子鼻传感器模块气敏传感器采用还原性气体传感器费加罗 TGS813 、TGS822 等，它们为 N 型半导体类气体传感器，其主要成份是二氧化锡烧结体。在正常工作温度下，吸附还原性气体时，因发生还原性气体的吸附与氧化反应，粒子界面存在的势垒降低，电子容易流动，从而电导率上升。当恢复到清洁空气中时，由于半导体表面吸附氧气，使粒子界面的势垒升高，阻碍电子的流动，电导率下降。将 TGS 传感器这种电导率变化，以输出电压的方式取出，从而可以检

测出气体的浓度。其优点是具有良好使用性和稳定性，以及低成本性。TGS813 型半导体气敏传感器非常适合检验甲烷、丙烷、丁烷等有毒气体； TGS822 可用于检测有机溶剂的蒸汽以及一些可燃气体如一氧化碳；TGS825 适用于硫化氢等有毒气体、TGS826 适用于胺类化合物尤其是氨气[45]。

参考 TGS813 气体敏感特性图，可知不同的 TGS813 的气敏电阻 Rs 变化范围大，电阻比值 R_s（3000×10^{-6}）/ R_s（1000×10^{-6}）=0.60±0.05，变化较小，对数坐标图上气体敏感特性基本为线性，空气中信号电阻很大，约10^4-10^5U [46]。

基于 TGS813 特性，对被测气体为甲烷气体时有：

$$k=\frac{\lg\frac{R_s}{R_o}}{\lg\frac{x}{1000}}=\frac{\lg(0.6\pm0.05)}{\lg\frac{3000}{1000}}=-0.392\sim-0.542 \tag{5.1}$$

k 为敏感系数； x 为气体浓度。

有上面的式子我们可以得到：

$$R_s=R_o(x/1000)^k \tag{5.2}$$

又有：

$$V_o=-\frac{V_{ref}}{R_s}R_f \tag{5.3}$$

通过公式（5.2）（5.3）推出：

$$V_o=-\frac{V_{ref}}{R_0}R_f\left(\frac{x}{1000}\right)^{-k} \tag{5.4}$$

通过公式（5.4）可以知道传感器电压输出与气体浓度直接关系，从而实现对甲烷气体浓度的检测。TGS813、TGS822 传感器应用电路[45]如图 5.2。

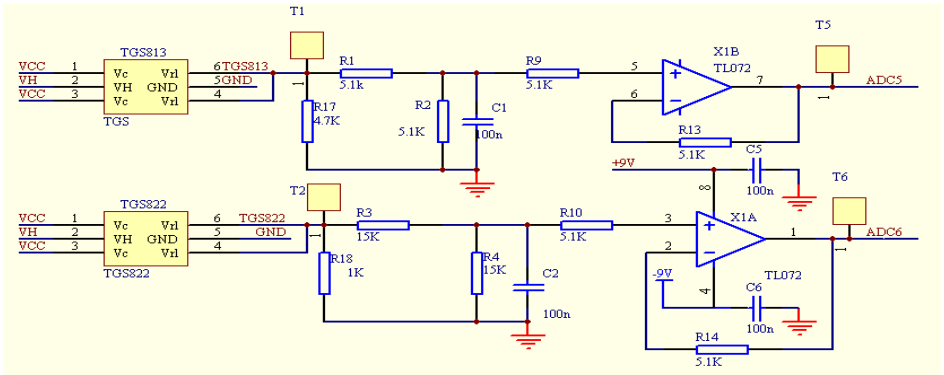

图 5.2 气敏传感器的典型应用电路

Fig.5.2 Gas sensor tpical application circuit

其中 TGS813、TGS822 的典型应用电路中，此处 TL072 芯片起到信号放大作用。

5.2 无线电子鼻传感器节点实现

如图 5.3 所示为无线电子鼻传感器节点，电路板为电子鼻传感器节点之传感器板，电源模块为传感器板在实验室内调试提供+9v，-9v，+5v 电源，同时传感器板上通过电压转换芯片实现+5v 到+3.3V 转换，从而实现对接在传感器板上的节点供电。

12Pin 接线端子其中有六路用于接传感器，从而把传感器与传感器节点 ADC 模块外部引脚相连。这六路传感器构成一个传感器阵列，从而最终实现所需感知气体的检测。

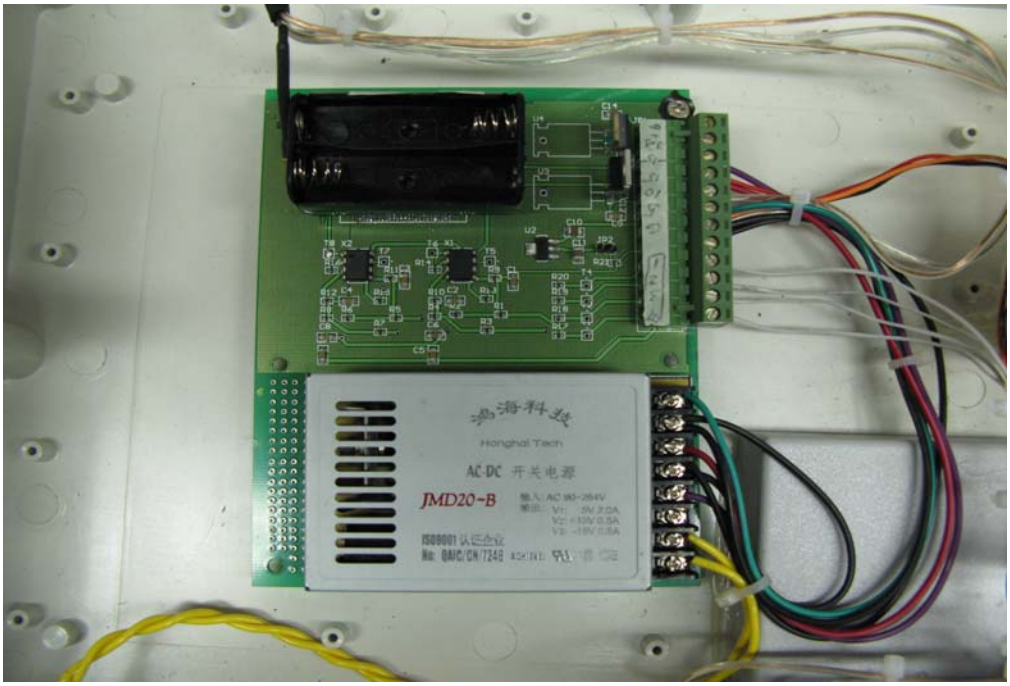

图 5.3 无线电子鼻传感器节点

Fig.5.3 Wireless E-nose sensor node

5.3 实验结果

本文实验环境是通过外部电源模块（非独立电池）为电子鼻节点、传感器供电。电子鼻节点传感器密封于容器中，可通过给密封容器加压灌气或阀门漏气，从而改变密封容器中的气体浓度。

实验结果数据主要通过接收来自 Sink 节点通过串口发送给 PC 机的数据包，图

5.4 中软件界面是使用 VC++6.0 开发工具编写的一个小程序，专用于接收来自 Sink 节点的数据。对于串口数据的接收主要通过 MSComm 控件，采用事件驱动方式实现每一次串口数据接收。

软件可以设置使用哪一个串口，并且可以通过该软件给某一节点发送数据。其中图 5.4 的右上角为实时从窗口过来的数据即 TinyOS 消息格式显示出来。图左侧为经过初步转化的电子鼻传感器节点采集数据，其中有六列主要信息 wsn_tgs1-wsn_tgs6，分别对应于我们使用六路传感器。

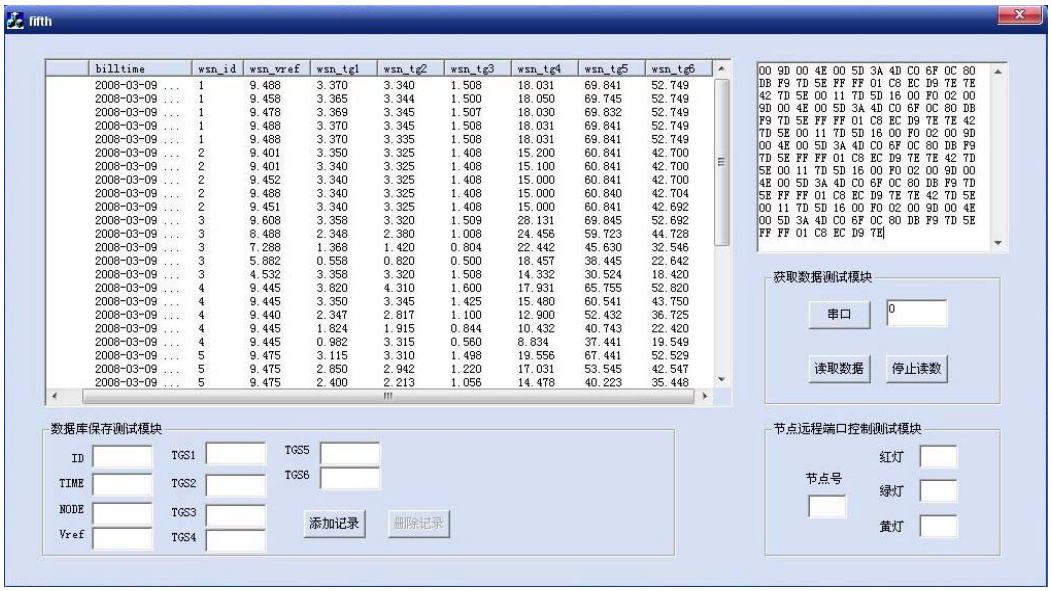

图 5.4 电子鼻传感器节点信息表

Fig.5.4 E-nose sensor node message table

实验通过 5 个电子鼻传感器采集节点，在一个大周期内，使每一个节点先后工作。而在每一个节点工作期间，发送周期为 2s，发送 5 次。见表 5.1 为截取电子鼻传感器节点采集信息表。

表 5.1 电子鼻传感器节点信息表

Table.5.1 E-nose sensor node message table

billtime	wsn_id	wsn_tg1	wsn_tg2	wsn_tg3	wsn_tg4	wsn_tg5	wsn_tg6
2008-03-09 21:45:40	1	3.370	3.340	1.508	18.031	69.841	52.749
2008-03-09 21:45:42	1	3.365	3.344	1.500	18.050	69.745	52.749
2008-03-09 21:45:44	1	3.369	3.345	1.507	18.030	69.832	52.749
2008-03-09 21:45:46	1	3.370	3.345	1.508	18.031	69.841	52.749
2008-03-09 21:45:48	1	3.370	3.335	1.508	18.031	69.841	52.749
2008-03-09 21:45:50	2	3.350	3.325	1.408	15.200	60.841	42.700
2008-03-09 21:45:52	2	3.340	3.325	1.408	15.100	60.841	42.700

续表 5.1

billtime	wsn_id	wsn_tg1	wsn_tg2	wsn_tg3	wsn_tg4	wsn_tg5	wsn_tg6
2008-03-09 21:45:54	2	3.340	3.325	1.408	15.000	60.841	42.700
2008-03-09 21:45:56	2	3.340	3.325	1.408	15.000	60.840	42.704
2008-03-09 21:45:58	2	3.340	3.325	1.408	15.000	60.841	42.692
2008-03-09 21:46:00	3	3.358	3.320	1.509	28.131	69.845	52.692
2008-03-09 21:46:02	3	2.348	2.380	1.323	24.456	59.723	44.728
2008-03-09 21:46:04	3	1.968	1.920	1.054	22.442	45.630	32.546
2008-03-09 21:46:06	3	1.534	1.554	0.840	18.457	38.445	22.642
2008-03-09 21:46:08	3	1.258	1.020	0.608	14.332	30.524	18.420
2008-03-09 21:46:10	4	3.820	4.310	1.600	17.931	65.755	52.820
2008-03-09 21:46:12	4	3.350	3.345	1.425	15.480	60.541	43.750
2008-03-09 21:46:14	4	2.347	2.817	1.100	12.900	52.432	36.725
2008-03-09 21:46:16	4	1.824	1.915	0.844	10.432	40.743	22.420
2008-03-09 21:46:18	4	1.282	1.325	0.560	8.834	37.441	19.549
2008-03-09 21:46:20	5	3.115	3.310	1.498	19.556	67.441	52.529
2008-03-09 21:46:22	5	2.850	2.942	1.220	17.031	53.545	42.547
2008-03-09 21:46:24	5	2.400	2.213	1.056	14.478	40.223	35.448
2008-03-09 21:46:26	5	1.754	1.945	0.920	10.745	37.123	26.649
2008-03-09 21:46:28	5	1.041	1.153	0.459	9.442	33.741	18.779

对于这 5 个节点，第 1、2 号节点用于检测静态稳定环境下的气体浓度，第 3、4、5 号节点用于观测通过逐步向密封容器内冲压气体从而气体浓度发生变化环境下的节点数据。观察这 5 个节点采集信息，发现第 1、2 节点数据信息稳定，只是局部发生了微小的变化。第 3、4、5 号随着向容器内不断的充气，表内数据值不断减小，说明气体浓读在不断减小。另外，通过多次实验，针对于静态稳定环境下测得传感器节点数据变化平均误差在%6.42 以内。

实验结果数据表明：电子鼻传感器节点工作稳定，数据变化出现在误差允许的范围内；功能正常，可以检测出气体浓度。

5.4 本章小结

本章节主要分析在实验室调试环境下基于 TGS 系列气敏传感器实现电子鼻传感器节点的实现，简述了实物的基本功能，通过实验测试，验证了电子鼻传感器节点功能和工作的稳定性。

6 结　论

随着无线传感器网络的不断深入研究和广泛应用，传感器网络已经逐渐深入到人类生活的各个领域。无线传感器网络节点是组成无线传感器网络的基本单位，是构成无线传感器网络的基础平台。传感器节点不仅要完成采集信息、融合并传送数据的功能，而且节点中的电源模块还负责节点的驱动，这是决定网络生存期的关键因素。因此节点技术的进步与无线传感器网络的发展有着密切的联系。

本文针对传感器节点硬件平台进行了模块化设计，实现了节点的硬件平台。在研究了源码开放的微型嵌入式 TinyOS 操作系统基础上，完成节点在软件程序设计，完成了基于气敏传感器的无线电子鼻传感器节点的实现。TinyOS 操作系统在软件体系结构上体现了轻量线程技术、主动消息通信技术、事件驱动模式、组件化编程，有助于提高无线传感器网络的性能，发挥硬件的特点，降低其功能，简化应用的开发。

课题研究完成了以下任务，并取得了相应的成果：

① 研究微型嵌入式 TinyOS 操作系统，深入了解了 TinyOS 组件模型、通信模型、事件驱动机制和并发模型以及 TinyOS 内核调度机制以及不足，更好的理解 TinyOS 是进行节点程序设计的基础。

② 在节点硬件设计方面：传感器节点模块化设计，完成了处理器模块、无线通信模块、电源模块和 Sink 节点接入接口的设计。采用 ATmega128L+CC2420 进行了 PCB 设计，完成了节点硬件设计上物理实现。

③ 在节点软件设计方面：软件模块化设计，在研究 TinyOS 的基础上完成了无线收发模块、ADC、串口等核心模块的程序实现，并在此基础上，实现了基于最小跳数算法的节点多跳通信。并探究了多跳路由中的链路质量估计和邻居表管理，以及在具体实现多跳通信过程中父节点选择、循环回路探测、重复数据包剔除等问题。

④ 完成了基于气敏传感器的无线电子鼻传感器节点的实现，验证了电子鼻传感器节点功能和工作的稳定性。

在本文设计中还有一些地方需要改进：

① 节点硬件方面的电源供应模块、节点体积需要在技术进一步改进。

② 在实现基于最小跳数多跳路由通信中的链路质量估计和邻居表管理在算法以及节点能量控制方面上进一步完善。

③ 相对于采用电池电源模块供电，TGS 系列气敏传感器一般体积比较大，功耗大。电子鼻功能与理想电子鼻功能还有很大差距，还需要不断研究。

致　　谢

本文的研究工作是在我的导师柴毅教授的精心指导和悉心关怀下完成的，在我的学业和论文的研究工作中无不倾注着导师辛勤的汗水和心血。导师的严谨治学态度、渊博的知识、无私的奉献精神使我深受的启迪。从尊敬的导师身上，我不仅学到了扎实、宽广的专业知识，也学到了做人的道理。在此我要向我的导师致以最衷心的感谢和深深的敬意。

在三年的学习生活中，还得到了学院众多老师、同学以及部队领导和同志们的热情关心和帮助。

在日常学习和生活中，我的师兄弟郭茂耘、曲剑锋、王道斌、李大杰、冯文武、张可、翁道磊等给予了我很大帮助。在此，向所有关心和帮助过我的领导、老师、同学和朋友表示由衷的谢意！

衷心地感谢在百忙之中评阅论文和参加答辩的各位专家、教授！

杨红远

二〇〇八年四月 于重庆

参考文献

[1] Wieser, Mark.The computer for the 21st Century[J].Scientific Americation,September 1991,265(3):66-75.

[2] 吴朝晖，潘纲.普适计算[C].中国计算机学会文集,清华大学出版社，北京 2006,8:175-185.

[3] 郑增威,吴朝晖.普适计算综述[J].计算机科学,2003,30(4)18-23.

[4] 孙利民,李建中等编著.无线传感器网络[M].北京 清华大学出版社.2005,5.

[5] IEEE Standards 802.15.4TM-2003,Wireless Medium Access Control (MAC) and Physical Layer(PHY) Specifications for Low-RateWireless Personal Area Networks(LR2WPANs).

[6] http://www.ieee802.org/15/pub/TG4.html.

[7] http://www.zigbee.org/en/press/.

[8] Hill J,Szewczyk R,Woo A,Hoolar S,Culler D E,Pister K . System Architecture Directions for Networked Sensors.In:Proc Architectual Support for Programming Languages and Operating System,2000,93~104.

[9] 马学童.室内空气品质监控系统的开发与研制.西北工业大学硕士学位论文.2002 年 3 月.

[10] 于鹏,潘敏,陈裕泉.一种基于全程动态扫描的白酒鉴别智能人工嗅觉系统.传感技术学报.2003 年 9 月第 3 期.

[11] Nanto H，et al. Identification of aromas from wine using quartz gas sensors in conjunction with neural-network analysis [J] .Sensors and Actuators B,1995,24-25;794-796.

[12] Bourrounet B,et al .Application of a multi-gas-sensor device in the meat industry for boar-taint detection [J].Sensors and Actuators B,1995,26-27;250-254.

[13] Gay D,Levis P,Culler D,Brewer E.nesC 1.1 Langue Refernce Manual,2003.

[14] Lynch C,O'reily F.Pic-based TinyOS implementation[C].Proceedings of the 2nd European Workshop Oil Sensor Networks.New York,USA:IEEE,2005:378-385.

[15] Levis P,Madden S,Polastre J,et aL.TinyOS:An operating system for wireless sensor networks[C/OL].WEBER W,RABAEY J,AARTS E.Ambient Intelligence.New York,NY：Springer-Verlag.2005.

[16] 林恺 赵海等.一种基于 TinyOS 的自适应双环调度策略[J].东北大学学报:自然科学版.2007,(28),7.

[17] 刘奎安,郭文生等 TinyOS 任务调度机制与实时调度构件设计[J].计算机应用.2007,(27),11.

[18] Paul Boone. SYSC 5701-Operating System Methods for Real-Time Applications. Real-Time Communication and Coordination in Wireless Embedded Sensor Networks.

[19] 沈建华，杨艳琴等编著.MSP430 系列 16 位超低功耗单片机原理与应用[M].清华大学出版

社.2005,8.

[20] ATmega128 中文手册.pdf.

[21] http://www.tinyos.net/hardware/telos/telos-revb-2004-09-27.pdf

[22] http://focus.ti.com/lit/ds/symlink/cc2420.pdf.

[23] http://www.ftdichip.com/Documents/DataSheets/DS_FT2232C.pdf

[24] 卢崇.无线传感器网络节点研制及网络管理软件开发[M].西北工业大学.2007,3.

[25] 王春雷,黄玉等.基于无线传感器网络的火灾监控系统设计与实现[J].计算机工程与设计.2007,(28),10.

[26] Shijin Dai,Xiaorong Jing .etc.Research and analysis on routing protocols for wireless sensor networks.IEEE Communications, Circuits and Systems, 2005. Proceedings. 2005 International Conference on.2005.5. 407– 411.

[27] 于磊磊，柴乔林等.基于最小跳数的无线传感器网络能量自适应路由算法.计算机应用研究.2007 年 11 月 24 卷 11 期.

[28] Y.Choi,M.G.Gouda,M.C.Kim,and A.Arora.The mote connectivity protocol.Technical Report TR-03-08, Department of Computer Sciences,TheUniversity of Texas at Austin,2003.

[29] P.G.Gibbons and Y.Matias.New sampling-basedsummary statistics for improving approximate query answers.In Proceedings of ACM SIGMOD International Conference on Management of Data,pages 311-342,June 1998.

[30] A.Adya,P.Bahl,J.Padhye,A.Wolman,and L.Zhou.A multi-radio unification protocol for IEEE802.11 wireless networks.In BroadNets.

[31] J. Yeo, H. Lee and S. Kim, An efficient broadcast scheduling algorithm for tdma ad-hoc networks, *Computer & Operations Research* 29 (2002), pp. 1793–1806.

[32] Alec Woo,Terence Tong,David Culler. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks[[EB/OL]]. http://www.cs.berkeley.edu/~awoo/sensys_awoo03.pdf

[33] J.Hill and D.Culler.Mica: a wireless platform for deeply embedded networks.IEEE Micro, 22(6):12–24,nov/dec 2002.

[34] D.De Couto,D.Aguayo,J.Bicket,and R.Morris.High-throughput path metric for multi-hop wireless routing.In MOBICOM,2003.

[35] Richard Draves,Jitendra Padhye,Brian Zill.Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks[EB/OL]. http://research.microsoft.com/mesh/papers/multiradio.pdf

[36] E.Shih,P.Bahl,and M.J.Sinclair. Wake on wireless:: an event driven energy saving strategy for battery operated devices. In Proceedings of the eighth annual international conferenc on Mobile computing and networking, pages 160–171. ACM Press, 2002.

[37] R.P.Draves,J.Padhye,and B.D.Zill.Comparisonof routing metrics for static multi-hop wirelessnetworks.In SIGCOMM,2004.

[38] A.Woo and D.Culler.Evaluation of Efficient Link Reliability Estimators for Low-Power Wireless Networks.Technical Report number to be assigned, University ofCalifornia, Berkeley, April 2003.

[39] R.Draves,J.Padhye,and B.Zill.The architecture of the Link Quality Source Routing Protocol.Technical Report MSR-TR-2004-57,Microsoft Research,2004.

[40] Z.Fu,P.Zerfos,H.Luo,S.Lu,L.Zhang,and M.Gerla.The Impact of Multihop Wireless Channelon TCP Throughput and Loss.In INFOCOM,2003.

[41] C.Intanagonwiwat,R.Govindan,and D.Estrin.Directed diffusion： a scalable and robust communication paradigm for sensor networks.In Proceedings of the International Conference on Mobile Computing and Networking Mobicom, pages 56-67,August 2000.

[42] C.Hedrick.Routing information protocol.In RFC1058,June 1988.

[43] Jerry Zhao,Ramesh Govindan.Understanding Packet Delivery Performance In Dense Wireless Sensor Networks. http://www.cens.ucla.edu/sensys03/proceedings/p1-zhao.pdf.

[44] 潘小青,黎定国.基于 P89LPC935 的可燃性气体检测报警系统的设计[J].江西理工大学学报.2007,(28),4.

[45] 张红剑,叶敦范等.可燃气体传感器 TGS813 在多路数据采集电路中应用[J]. 国外电子元器件.2007,11.

[46] 武贺恩.电子鼻技术在电气火灾监测系统中的应用研究[D].浙江大学.2007.4.

附　　录

A.作者在攻读硕士学位期间发表的论文目录

[1] 杨红远,柴毅,屈剑锋,郭茂耘.火箭发射测控中目标跟踪系统.电子技术应用.2007 年第 12 期 134~136.

独 创 性 声 明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 重庆大学 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：杨尔远　　签字日期：2008 年 6月 4 日

学位论文版权使用授权书

本学位论文作者完全了解 重庆大学 有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权 重庆大学 可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

保密（　），在____年解密后适用本授权书。

本学位论文属于

不保密（√）。

（请只在上述一个括号内打“√”）

学位论文作者签名：杨尔远　　导师签名：

签字日期：2008 年 6 月 4 日　　签字日期：2008 年 6 月 4 日

CHONGQING UNIVERSITY