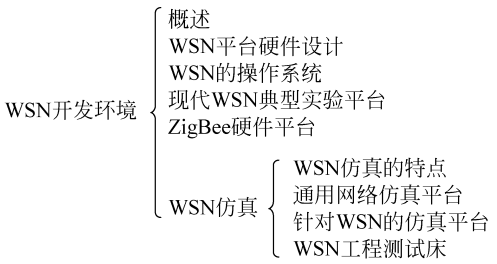
第 3 章 WSN开发环境

学习导航



学习目标

- 了解 WSN 节点设计模块设计内容、WSN 节点开发案例、WSN 中常见节点内容、TinyOS 操作系统、nesC 语言、LED 灯闪烁实验、WSN 实验平台中传感器节点内容、CC2530 片上内核、CC2530 主要特征外设、CC2530 无线收发器、WSN 仿真中的各种平台。
- 掌握 WSN 系统结构图、WSN 节点设计内容、WSN 节点设计要求、WSN 实验平台的总体结构图、WSN 实验平台中的仿真器原理、WSN 实验平台中的路由节点内容、CC2530 芯片的特点和 CC2530 开发环境 IAR。

3.1 WSN 概述

国内目前在无线传感器网络软件、硬件方面都在相应地发展，在基于国际标准、操作系统之上，许多公司都已研发了自己的硬件平台、中间件软件。武汉创维特信息技术有限公司的 CVT-WSN-S 全功能无线传感器网络实验平台，深圳市无线龙科技有限公司的 C51RF-WSN 无线传感器网络开发平台，提供了功能齐全的硬件开发环境，对外提供便捷的接口，用户无须了解底层细节，极大地降低了无线传感器应用的开发难度。

无线 ZigBee 传感器网络系统主要由计算机、网关和网络节点等组成。用户可以很方便地实现传感器网络无线化、网络化、规模化的演示、观测和进行二次开发。

(1) 计算机部分，主要完成接收网关数据和发送指令，实现可视化、形象化人机界面，方便用户操作和观察。

(2) 网关部分，主要完成通过计算机进行的指令发送或接收路由节点或者传感器节点数据，并将接收到的数据发送给计算机。

(3) 路由节点部分，主要在网关不能和所有的传感器节点通信时，路由节点作为一种中介使网关和传感器节点通信，实现路由通信功能。

(4) 传感器节点，主要完成对设备的控制和数据的采集，如灯的控制温度、光照度数据等。

3.2 WSN 平台硬件设计

▶ 3.2.1 系统结构图

现代无线传感器网络开发，各种实验平台根据不同的情况可以由一台计算机、一个网关、一个或多个路由器、一个或多个传感器节点组成。系统大小只受 PC 软件观测数量、路由深度、网络最大负载量限制。

现代无线传感器网络开发，各种实验平台内均配置 ZigBee2007/PRO、ZigBee2006 等协议栈，典型的传感器网络系统结构如图 3-1 所示。

基于 ZigBee2007/PRO、ZigBee2006 等协议栈的无线传感器网络具有自组织能力，其在网络设备安装和架设过程中自动完成网络组建。完成网络的架设后，用户便可以由 PC 发出命令，读取网络中任何设备上挂接的传感器的数据以及测试其电压，其简单的工作流程描述如图 3-2 所示。

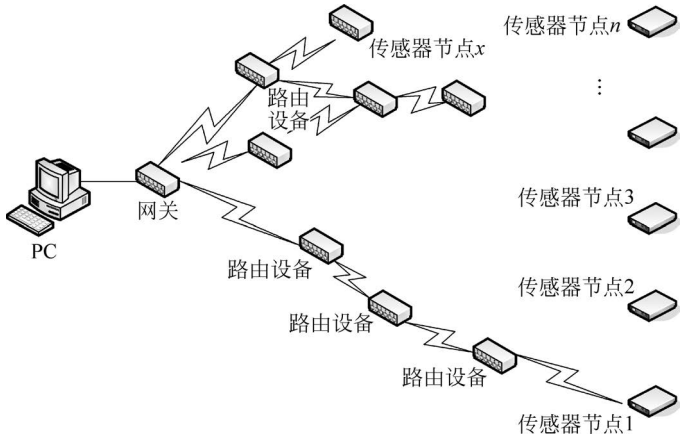


图 3-1 传感器网络系统结构

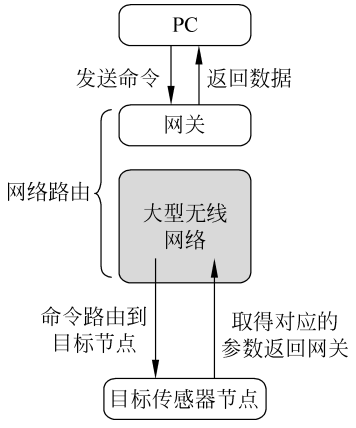


图 3-2 工作流程

同时，在使用无线 ZigBee 传感器网络实验平台开发应用系统前，需要学习理解一些基础知识，如无线数据传输、ZigBee 无线网络和传感器知识。

▶ 3.2.2 节点设计内容与要求

1. 节点的设计要求

根据应用环境的不同，传感器网络对节点的精度、传输距离、使用频段数据收发效率和功耗等提出了不同的要求，要求搭建相应的硬件系统和软件系统，使节点能够持续、可靠和有效地工作，其传感器节点的设计主要有以下 5 点要求。

(1) 微型化。无线传感器节点在体积上应足够小，以保证对目标系统本身的特性不造成显著影响。在某些应用场合，如战场侦察，甚至需要节点体积小到不容易让人察觉的程度，以完成一些特殊任务。

(2) 低功耗。节点部署后需要长期在建筑物内或野外等环境工作，携带电量有限，电池更换可行性较低，必须具备低功耗的性能。在硬件设计方面，电路应尽可能简单实用，尽可能选择低功耗器件。

（3）低成本。无线传感器网络由大量密集分布的节点组成，只有低成本才有可能大量地布置在目标区域中。低成本对传感器部件提出苛刻的要求。首先，供电模块必须简单且造价低；其次，能量有限，要求所有的器件必须是低功耗的；最后，传感器不能使用精度过高的部件，以免造成传感器模块成本过高。

（4）稳定性和安全性。节点的各个部件应该能够在给定的外部变化范围内正常工作。在给定的温度、湿度、压力等外部条件下，无线传感器网络节点的处理器、无线通信模块和电源模块要保证正常的功能，并使感知部件工作于各自的量程范围内。节点在恶劣的环境下要能稳定工作，且要有数据完整性保护，以防止外界因素造成的数据变化。

（5）扩展性和灵活性。无线传感器网络节点需要定义统一完整的外部接口，以便必要时在现有节点上直接添加新的硬件功能模块，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自由组合。在不同的应用环境下选择不同的组件配置系统，无须为每个应用都开发一套新的硬件系统。当然，部件的扩展性和灵活性应该以保证系统的稳定性为前提，必须考虑连接器件的性能。

2. 节点硬件设计内容

大多数传感器网络节点具有终端探测和路由的双重功能：一方面实现数据的采集和处理；另一方面实现数据的融合和路由，对本身采集的数据和收到的其他节点发送的数据进行综合，转发路由到网关节点。网关节点往往个数有限，而且能量常常能够得到补充。网关通常使用多种方式（如 Internet、卫星或移动通信网络等）与外界进行通信。

传感器节点的硬件平台结构如图 3-3 所示。传感器节点一般由数据处理器模块、存储器模块、无线通信模块、传感模块和电源模块 5 部分组成。数据处理模块是节点的核心模块，用于完成数据处理、数据存储、执行通信协议和节点调度管理等工作；存储器模块主要完成存储处理器转送的数据；无线通信模块主要完成在信道上发送和接收信息；传感器模块主要采集监控或观测区域内的物理信息；电源模块主要为各个功能模块提供能量。

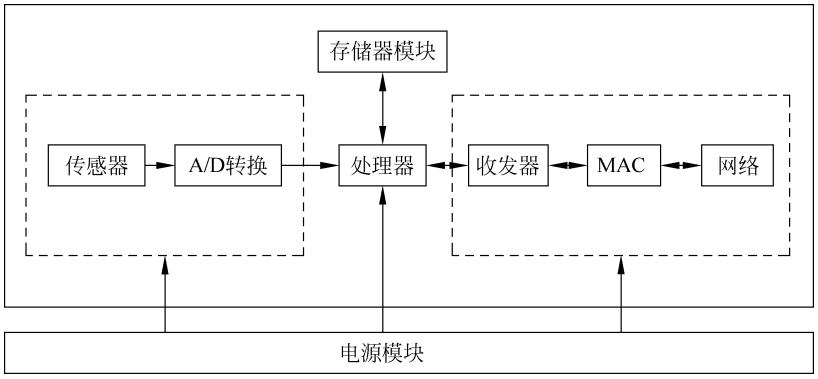


图 3-3　传感器节点的硬件平台结构

▶ 3.2.3　节点的模块化设计

1. 处理器模块

处理器模块是无线传感器网络节点的核心部件，微处理器选型应满足以下 4 方面的要求。

（1）体积尽量小，处理器的尺寸基本决定整个节点的尺寸。

（2）集成度尽可能高，要有足够的外部通用 I/O 接口和通信接口，使整个系统的处理器外

围电路简单整洁，基本不需要扩展额外的器件，减小整个节点的尺寸。

(3) 功耗低且支持休眠模式，休眠模式直接关系到节点生命周期的长短，系统在绝大多数时间内应处于待机或休眠状态。

(4) 运行速度快，系统能够在最短的时间内完成工作，进入休眠状态，节省系统能量。

从处理器的角度来看，传感器网络节点基本可以分为三类。

第一类采用以 ARM 处理器为代表的高端处理器。该类节点的能量消耗比采用微控制器大很多，多数支持 DVS(电压调节)或 DFS(动态频率调节)等节能策略，但是其处理能力也强很多，适合图像等高数据量业务的应用。另外，采用高端处理器来作为网关节点也是不错的选择。ARM(Advanced RISC Machines)处理器是一种高端处理器，除了具备 RISC 体系结构的优点外，还采用了一些特别的技术，在保证高性能的前提下尽量缩小芯片的面积并降低功耗。ARM 处理器具有多个系列，除了具有 ARM 体系结构的共同特点外，每个系列的 ARM 处理器都具有各自的特点。常见的 7 个系列包括 ARM7 系列、ARM9 系列、ARM9E 系列、ARM10E 系列、SecurCore 系列、Intel 的 XScale 系列和 Intel 的 Strong ARM 系列。其中，ARM7、ARM9、ARM9E 和 ARM10E 为 4 个通用处理器系列，每一个系列都有相应的独特性能。SecurCore 系列专为安全性要求较高的应用设计，Intel 的 XScale 为体系结构提供一种新的高性价比、低功耗的解决方案，支持 16 位 Thumb 指令和 DSP 扩充。

第二类是以采用低端微控制器为代表的节点。该类节点的处理能力较弱，但是能量消耗也很小。在选择处理器时应该首先考虑系统对处理能力的需要，然后再考虑功耗问题。低端处理器主要有 Atmel 公司的 AVR 系列单片机和 TI 公司的 MSP430 系列单片机，它们是目前传感器网络领域应用较多的低端处理器，其共同点是超低功耗、具有完整的外部接口及较高的集成度。AVR 系列处理器采用 RISC(Reduced Instruction Set Computer)结构，吸取了 PIC 和 8051 等系列单片机的优点，在内部结构上进行了较大改进。MSP430 单片机内核是 16 位 RISC 处理器，单指令周期，其运算能力和速度具有一定的优势。作为超低功耗处理器，MSP430 系列单片机具有 5 种不同深度的低功耗休眠模式。其中，MSP430F1 系列处理器工作电压为 1.8V，待机工作电流为 1mA，工作在 1MHz 频率时的电流为 300μA。

第三类是数字信号处理器，DSP(Digital Signal Processor)是一种独特的微处理器，适合实时数字信号处理。除了具备普通处理器所拥有的运算和控制能力外，DSP 针对实时信号处理的特点，在处理器的结构、指令系统、指令流程上做了较大改进。其工作原理是接收模拟信号并转换为 0 或 1 的数字信号，再对数字信号进行修改、删除、强化，并在其他系统芯片中把数字数据转换为模拟数据或实际环境格式。它不仅具有可编程性，而且实时运行速度可达每秒数以千万条复杂指令程序，远远超过通用微处理器。由于适合处理大批量的数据，在传感器网络中经常利用 DSP 处理无线通信设备发出的信息，提取数据流，对测量信息进行节点内处理等。

微处理器单元是传感器网络节点的核心，负责整个节点系统的运行管理。各种常见的微控制器性能比较如表 3-1 所示。

表 3-1　各种常见的微控制器性能比较

厂商	芯片型号	RAM 容量/KB	Flash 容量/KB	正常工作电流/mA	休眠模式下的电流/μA
Atmel	Mega103	4	128	5.5	1
	Mega128	4	128	8	20
	Mega165/325/645	4	64	2.5	2

续表

厂商	芯片型号	RAM 容量/KB	Flash 容量/KB	正常工作电流/mA	休眠模式下的电流/μA
Microchip	PIC16F87x	0.36	8	2	1
Intel	8051 8 位 Classic	0.5	32	30	5
	8051 16 位	1	16	45	10
Philips	80C51 16 位	2	60	15	3
Motorola	HC05	0.5	32	6.6	90
	HC08	2	32	8	100
	HCS08	4	60	6.5	1
TI	MSP430F14x16 位	2	60	1.5	1
	MSP430F16x16 位	10	48	2	1
Atmel	AT91 ARM Thumb	256	1024	38	160
Intel	XScale PXA27x	256	N/A	39	574
Samsung	S3C44B0	8	N/A	60	5

在选择处理器时，应该首先考虑系统对处理能力的需要，然后考虑功耗问题。不过对于功耗的衡量标准不能仅从处理器有几种休眠模式、每兆赫兹时钟频率所耗费的能量等角度去考虑处理器自身的功耗，还要从处理器每执行一次指令所耗费的能量这个指标综合考虑。表 3-2 是目前一些常用处理器在不同的运行频率下每执行一次指令所耗费能量的数据列表。

表 3-2　常用处理器每执行一次指令所耗费的能量

芯片型号	运行电压/V	运行频率	单位指令消耗能量/nJ
ATMega128L	3.3	4MHz	4
ARM Thumb	1.8	40MHz	0.21
C8051F121	3.3	32kHz	0.2
IBM 405LP	1	152MHz	0.35
C8051F121	3.3	25MHz	0.5
TMS320VC5510	1.5	200MHz	0.8
Xscale PXA250	1.3	400MHz	1.1
IBM 405LP	1.8	380MHz	1.3
Xscale PXA250	0.85	130MHz	1.9

目前处理器模块中使用较多的是 Atmel 公司的单片机。它采用 RISC 结构，吸取了 PIC 和 8051 单片机的优点，具有丰富的内部资源和外部接口。在集成度方面，其内部集成了几乎所有关键部件；在指令执行方面，微控制单元采用 Harvard 结构，因此指令大多为单周期；在能源管理方面，AVR 单片机提供多种电源管理方式，尽量节省节点能量；在可扩展性方面，提供多个 I/O 口，并且和通用单片机兼容；此外，AVR 系列单片机提供的 USART（通用同步异步收发器）控制器、SPI（串行外围接口）控制器等与无线收发模块相结合，能够实现大吞吐量、高速率的数据收发。

TI 公司的 MSP430 超低功耗系列处理器，不仅功能完善、集成度高，而且根据存储容量的多少提供多种引脚兼容的系列处理器，使开发者可以根据应用对象灵活选择。

另外，作为 32 位嵌入式处理器的 ARM 单片机，也已经在无线传感器网络方面得到了应用。如果用户可以接受它的较高成本，那么可以利用这种单片机来运行复杂的算法，完成更多的应用业务功能。

2. 存储模块

存储器主要包括随机存储器(RAM)和只读存储器(ROM)。RAM 可以分为 SRAM、DRAM、SDRAM、DDRAM 等几类；ROM 又可分为 NOR Flash、EPROM、EEPROM、PROM 等几类。RAM 存储速度较快，但断电后会丢失数据，一般用于保存即时信息，如传感器的即时读入信息、其他节点发送的分组数据等。程序代码一般存储于只读存储器、电可擦除可编程只读存储器(EEPROM)或闪存中。存储器的选择应根据具体情况决定，通常根据成本和功耗来衡量，由于 RAM 成本和功耗较大，在设计传感器节点时应尽量减少 RAM 的大小。

3. 无线通信模块

无线通信模块由无线射频电路和天线组成，目前采用的传输介质主要包括无线电、红外、激光和超声波等，它是传感器节点中最主要的耗能模块，是传感器节点的设计重点。

现今传感器网络应用的无线通信技术通常包括 IEEE 802.11b、IEEE 802.15.4(ZigBee)、Bluetooth、UWB、RFID 和 IrDA 等，还有很多芯片双方通信的协议由用户自己定义，这些芯片一般工作在 ISM(Industrial Scientific Medical)免费频段。表 3-3 为目前传感器网络应用的常见无线通信技术列表。

表 3-3　传感器网络的常用无线通信技术

无线技术	频　　率	距离/m	功耗	传输速率/(kb·s^{-1})
Bluetooth	2.4GHz	10	低	10 000
802.11b	2.4GHz	100	高	11 000
RFID	50kHz～5.8GHz	＜5	—	200
ZigBee	2.4GHz	10～75	低	250
IrDA	0.3～400THz	1	低	16 000
UWB	3.1～10.6GHz	10	低	100 000
RF	300～1000MHz	10^X～100^X	低	10^X

注：X 表示数字 1～9

在无线传感器网络中应用最多的是 ZigBee 和普通射频芯片。ZigBee 是一种近距离、低复杂度、低功耗、低数据速率、低成本的双向无线通信技术，完整的协议栈只有 32KB，可以嵌入各种微型设备中，同时提供地理定位功能。

对于无线通信芯片的选择问题，从性能、成本和功耗方面考虑，RFM 公司的 TR1000 和 Chipcon 公司的 CC1000 是理想的选择。这两种芯片各有所长，TR1000 功耗低些，CC1000 灵敏度高些、传输距离更远。WeC、Renee 和 Mica 节点均采用 TR1000 芯片；Mica2 采用 CC1000 芯片；Mica3 采用 Chipcon 公司的 CC1020 芯片，传输速率可达 153.6kb/s，支持 OOK、FSK 和 GFSK 调制方式；3MicaZ 节点则采用 CC2420 ZigBee 或者 CC2530 ZigBee 芯片。

另外有一类无线芯片本身集成了处理器，例如，CC2430 是在 CC2420 的基础上集成了 51 内核的单片机；CC1010 是在 CC1000 的基础上集成了 51 内核的单片机，使得芯片的集成度进一步提高。WiseNet 节点就采用了 CC1010 芯片。常见的无线芯片还有 Nordic 公司的 nRFg05、nRF2401 等系列芯片。传感器网络节点常用的无线通信芯片的主要参数如表 3-4 所示。

表 3-4　常用射频芯片的主要参数

芯片/参数	频段/MHz	速率/(kb·s^{-1})	电流/mA	灵敏度/dBm	功率/dBm	调制方式
TR1000	916	115	3	－106	1.5	OOK/FSK
CC1000	300～1000	76.8	5.3	－110	20～10	FSK

续表

芯片/参数	频段/MHz	速率/(kb·s^{-1})	电流/mA	灵敏度/dBm	功率/dBm	调制方式
CC1020	402～904	153.6	19.9	－118	20～10	GFSK
CC2420	2400	250	19.7	－94	－3	O～QPSK
nRF905	433～915	100	12.5	－100	10	GFSK
nRF2401	2400	1000	15	－85	20～0	GFSK
9Xstream	902～928	20	140	－110	16～20	FHSS

目前市场上支持 ZigBee 协议的芯片制造商有 Chipcon 公司和 Freescale 半导体公司。TI/Chipcon 公司的 CC2420/CC2530 芯片应用较多，该公司还提供 ZigBee 协议的完整开发套件。Freescale 半导体公司提供 ZigBee 的 2.4GHz 无线传输芯片，包括 MC13191、MC13192、MC13193，该公司也提供配套的开发套件。

在无线射频电路设计中，主要考虑以下三个问题。

1）天线设计

在传感器节点设计中，根据不同的应用需求选择合理的天线类型。天线的设计指标有很多种，无线传感器网络节点使用的是 ISM/SRD 免证使用频段，主要从以下三个指标来衡量天线的性能。

（1）天线增益是指天线在能量发射最大方向上的增益，当以各向同性为增益基准时，单位为 dBi；如果以偶极子天线的发射为基准，单位为 dBd。天线的增益越高，通信距离就越远。

（2）天线效率是指天线以电磁波的形式发射到空中的能量与自身消耗能量的比值，其中，自身消耗的能量是以热的形式散发的。对于无线节点来说，天线辐射电阻较小，任何电路的损耗都会较大程度地降低天线的效率。

（3）天线电压驻波比主要用来衡量传输线与天线之间阻抗失配的程度。天线电压驻波比值越高，表示阻抗失配程度越高，则信号能量损耗越大。

天线的种类主要有以下三种。

（1）内置天线由于便于携带，且具有免受机械和外界环境损害等优点，常常是设计时的首选方案。其优点是成本低，缺点是性能较差。

（2）将简单的导线天线或金属条带天线作为元件，安装在电路板上。这种天线因损耗很低，并置于电路板上方，比印刷天线的通信性能有明显提高。导线天线是介于低成本、低效率的印刷天线与相对高成本、高效率的外置天线之间的一种很好的折中天线方案。

（3）外置天线通常没有内置天线那样的尺寸限制，通常离节点中的噪声源的距离较远，因而具有很高的无线通信传输性能。对那些需要尽可能最大的距离、必须选用定向天线的应用来说，外置天线几乎是必选的。

2）阻抗匹配

射频放大输出部分与天线之间的阻抗匹配情况，直接关系到功率的利用效率。如果匹配不好，很多能量会被天线反射回射频放大电路，不仅降低了发射效率，严重时还会导致节点的电路发热，缩短节点寿命。由于传感器节点通常使用较高的工作频率，因而必须考虑导线和 PCB 基板的材质、PCB 走线、器件的分布参数等诸多可能造成失配的因素。

3）电磁兼容

由于传感器节点体积小，包括微处理器、存储器、传感器和天线在内的各种器件，它们聚集在相对狭小的空间，因而任何不合理的设计都可能带来严重的电磁兼容问题。例如，由于天线

辐射造成传感器的探测功能异常，或微处理器总线上的数据异常等。

由于高频强信号是造成电磁兼容的主要原因，所以包括微处理器的外部总线、高速 I/O 端口、射频放大器和天线匹配电路等是电磁兼容设计中考虑的主要因素。

4. 传感器模块

根据实际需求可以选择具体的传感器节点实现数据采集功能。在传感器网络中，传感器的选择除了要考虑基本的灵敏度、线性范围稳定性及精确度等静态特性，还要综合功耗、可靠性、尺寸和成本等因素。传感器网络与传统传感器相比具有很多优点，如传感器网络可以将分布式的信息集中起来进行综合分析，减少单个测量造成的瞬态误差和单点激变造成的测量误差，使信息更加可靠和准确；网络化处理降低了对单个传感器的要求，利用区域内的多个测量数据，通过统计方法可以得到更高精度的数据；通过网络可以得到大量数据，这使技术人员将更多的精力集中在数据处理上，提高工作效率。

5. 电源模块

电源模块作为无线传感器网络的基础模块，直接关系到传感器节点的寿命、成本和体积，因此，在设计电源时主要应考虑以下三个方面的问题。

1）能量供应

电池供电是目前最常见的传感器节点供电方式。电池的主要量度指标是能量密度，即 J/cm^3。

电池的主要性能指标包括：

（1）标称电压。指单节新电池（电量充足时）的输出电压。

（2）内阻。电池内作为电解质的电解液存在一定的电阻，称为电池的内阻，当负载电流较大时，内阻压降会导致电池输出电压下降。

（3）容量。将放电电流与放电电压的乘积作为电池容量单位，单位一般用 mA·h（毫安·时）或 A·h（安·时）表示。标称容量为 1000mA·h 的理想电池，能够在 1000mA 的电流下工作 1h，或在 100mA 电流下工作 10h，实际上，随着工作电流的增大容量会下降。

（4）放电终止电压。当电压下降到终止电压时，说明电池耗尽，放电终止电压与标称电压越接近，说明电池放电越平稳。若系统中具有电量不足报警功能，报警值一般取略高于终止电压值。

（5）自放电。随着电池存储时间的增加，电解质和电极活性材料会逐渐消失，容量也会下降。如某电池存储年限为 5 年，则该电池容量会在 5 年内下降至 80%。传感器节点一般长时间不更换电池，因此应选择自放电较缓慢的电池。

（6）使用温度。电池内有液体或凝胶状电解液，环境温度过高或过低会导致电解质失效。相对于传感器节点的其他部件，电池的工作温度范围较窄。

电池可分为不可充电电池（原电池）和可充电电池（二次电池），充电电池能够实现能量补充，电池内阻较小。但是，它的不足之处是能量密度有限，质量能量密度较大，自放电问题严重。

表 3-5 给出了常用电池的性能参数。

表 3-5　常用电池的性能参数

电池类型	质量/能量/($W \cdot h \cdot kg^{-1}$)	体积/能量/($W \cdot h \cdot L^{-1}$)	循环寿命/次	工作温度/℃	内阻/mΩ
镍铬	41	120	500	20～60	7～19
镍氢	50～80	100～200	800	20～60	18～35
锂离子	120～160	200～280	1000	0～60	80～100
聚合物	140～180	＞320	1000	0～60	80～100

锂电池是目前发展最快、应用最广泛的电池之一，具有重量轻、容量大、性能优异等特点。

锂离子电池是一种可充电的锂电池，标称电压为4.2V，终止放电电压为3.7V。其放电过程相对平稳，没有记忆效应，且剩余容量与电压基本呈线性关系，自放电较小，一次充电可以存储较长时间(2年以上)。锂-亚硫酰氯电池是一种特种锂电池，标称电压为3.6V，具有较高的工作温度范围。在常温中，放电曲线较为平坦；在－40°C的低温环境下可以维持常温容量的50%左右；在120°C的环境下，若其年自放电为2%左右，则存储时间可达10年以上。锂-亚硫酰氯电池分为高容量型和高功率型两种，高容量型适合小电流长期放电，容量大，内阻也较大；高功率型能够提供较大放电电流，但容量较低。

2) 能量获取

一旦能量耗尽节点就会失效，为了延长节点和传感器网络的工作寿命，必须考虑从节点所处的环境中获取能量并为节点所用。常用能量产生方式主要有以下三种。

(1) 太阳能。太阳能电池能够为传感器节点供电，有效功率取决于使用环境(室内或室外)和使用时间。在户外环境下，输出功率约为10μW/cm^2，在室内环境下约为15μW/cm^2。单块电池提供的稳定输出电压约为0.6V。

(2) 温度梯度。温差可直接转换为电能。5K温差即可产生80μW/cm^{2}的输出功率和1V的输出电压。

(3) 振动。基于电磁学、静电学或压电学原理可以将机械能转化为电能，微机电系统(Micro-Electron Mechanical Systems，MEMS)即可将振动转换为电能。对于2.25m/s^2、120Hz的振动源，体积为1cm^{2}的MEMS装置可以产生大约200μW/cm^{2}的能量，足够向简单收发机供能。

3) 直流-直流转换

节点所需要的电压通常不是一种，而且随着电池使用时间的增加，容量会随之减少，电压也会降低。供电功率的降低会影响晶振频率和传输功率，通过限制供给节点电路的电压，利用直流-直流转换器可以克服这些问题。

直流-直流转换器有以下三种类型。

(1) 线性稳压开关，产生较输入电压低的电压。

(2) 开关稳压器，能升高电压、降低电压或翻转输入电压。

(3) 充电泵，可以升压、降压或翻转输入电压，但驱动能力有限。此外，直流-直流转换器自身也会消耗能量，所以会降低整体的效率，直流-直流转换器的工作效率也是设计中需要考虑的因素。

线性稳压器体积小、价格低、噪声小，其输入/输出使用退耦电容过滤，该电容不但有利于平稳电压，而且有利于去除电源中的瞬间短时脉冲波形干扰。许多嵌入式模块包括检电器，电源的瞬间变弱会严重影响系统的正常运行。若输入电压过低，检电器会重新启动处理器。

开关稳压器是具有高输入阻抗、低开关速度及低功耗的开关功率管，在变换输入电压为输出电压时，开关稳压器的功耗更低、效率更高。其缺点是需要较多的外部器件，需要占用较大的空间，而且开关稳压器比线性稳压器价格高，噪声也较大，但是功能比线性稳压器强大。与开关稳压器相似，充电泵能够升压、降压和翻转输入电压，但是其电流供应能力有限。

▶ 3.2.4　传感器节点开发实例

传感器节点的设计需要经过很多步骤，其流程图如图3-4所示，其中还需要很多重复工

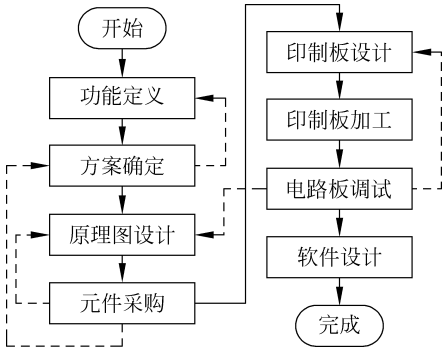


图 3-4　传感器节点设计流程图

作，虚线表示如果某个步骤不能顺利完成，则需要进行反复工作。在此以一个采用 Atmel 公司的 AVR 系列单片机和 Chipcon 公司的 CC2420 无线收发器的传感器节点设计为例来介绍传感器节点的设计过程，在此不考虑电源部分和传感器部分的设计。

1. 功能分析和芯片选型

根据传感器节点的特性，要求系统功耗相当低、体积小，而对传输距离和传输速率没有太多限制，考虑到 ZigBee 技术的特点，选用支持 ZigBee 的无线收发器。另外，为了系统使用方便，采用模块化设计，将电源部分、传感器部分做成独立的板子，而将单片机和无线收发器做成一块板子。在此选择使用 Atmel 公司的 AVR 系列单片机 ATMEGA128 和 Chipcon 公司的 CC2420 无线收发器。

ATMEGA128 的特点如下。

- AVR RISC 指令集，123 条指令，大部分为单周期指令。
- 32 个通用寄存器，64 个 I/O 控制寄存器。
- 片内 128KB 程序存储器，4KB 数据存储器（可外扩至 64KB）。
- 在线可编程。
- 片内模拟比较器。
- 可编程 UART、I^2C、SPI 接口。
- 可编程 RTC、看门狗、时钟源及计数器。
- 8 通道 10 位 ADC。
- 可编程 PWM。
- 层次化功耗管理。
- 6 种低功耗模式。
- 软件可配置的多时钟模式。
- 极低的功耗（正常功能下小于 5mW，低功耗模式下小于 10μW）。
- 采用 0.181μm 工艺。

CC2420 是 Chipcon 公司推出的首款符合 2.4GHz IEEE 802.15.4 标准的无线收发器。它基于 Chipcon 公司的 SmartRF03 技术，以 0.18μm CMOS 工艺制成，只需极少外部元器件，性能稳定且功耗极低。CC2420 的选择性和敏感性指数超过了 IEEE 802.15.4 标准的要求，可确保短距离通信的有效性和可靠性。利用此芯片开发的无线通信设备支持数据传输率高达 250kb/s，可以实现多点对多点的快速组网。CC2420 的特点如下。

- 工作频带范围：2.400～2.4835GHz。
- 采用 IEEE 802.15.4 规范要求的直接序列扩频方式。
- 数据速率达 250kb/s，码片速率达 2MChip/s。
- 采用 O-QPSK 调制方式。
- 超低电流消耗（RX：19.7mA，TX：17.4mA），高接收灵敏度（－99dBm）。
- 抗邻频道干扰能力强（39dB）。
- 内部集成 VCO、LNA、PA 以及电源整流器，采用低电压供电（2.1～3.6V）。
- 输出功率编程可控。

- IEEE 802.15.4 MAC层硬件可支持自动帧格式生成、同步插入与检测、16b CRC校验、电源检测、完全自动MAC层安全保护(CTR,CBC-MAC,CCM)。
- 与控制微处理器的接口配置容易(4总线SPI接口)。
- 开发工具齐全,提供开发套件和演示套件。
- 采用QLP-48封装,外形尺寸只有7mm。

选择了芯片之后需要对系统的总体结构进行设计,了解各部分之间的通信方式,目前设计的节点结构如图3-5所示。

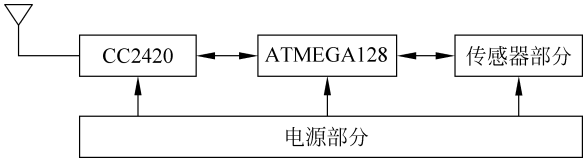


图3-5　自行设计的节点结构

2. 原理图及印制板设计

在芯片选型及硬件结构确定之后,需要对原理图进行设计,将一个个元器件按一定的逻辑关系连接起来。原理图设计需要执行以下步骤。

(1) 确定设计图纸大小。

(2) 设置设计环境。

(3) 布放元器件。

(4) 原理图布线、布线优化。

(5) 输出报表。

(6) 打印或保存文件。

在此仅介绍处理器的时钟部分、处理器与无线模块的接口部分和无线收发器三部分的原理图设计。

1) 处理器时钟电路设计

为了节约系统能源,在执行一些比较复杂的运算时,系统采用高频率时钟,而在其他时刻采用低频率时钟,甚至将系统时钟关闭使系统处于休眠状态。因此,处理器需要设计两个时钟频率的时钟电路,其电路原理图如图3-6所示。

2) 处理器无线模块接口设计

处理器模块与无线收发器模块之间的连接非常简单,CC2420使用SFD、FIFO、FIFOP和CCA 4个引脚表示收发数据的状态,处理器通过SPI与CC2420交换数据,发送命令,它们之间的逻辑连接图如图3-7所示。

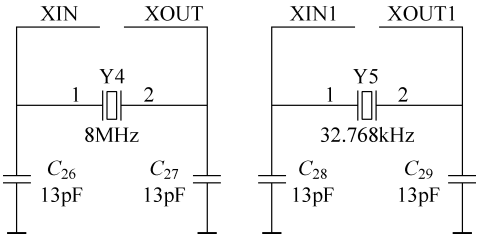


图3-6　时钟电路图

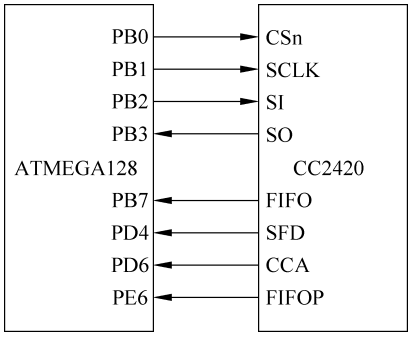


图3-7　ATMEGA128与CC2420接口

3）无线收发器模块设计

CC2420 只需要极少的外围元器件，它的外围电路包括晶振时钟电路、射频输入/输出匹配电路和微控制器接口电路三部分。

原理图设计完成之后要进行网表输出，并进行印制电路板设计，在设计过程中要特别注意无线收发器部分的布线和 PCB 天线的设计，这是传感器节点设计的难点和重点。在印制电路板设计完成之后将印制板文件送至印制板制作厂家生产。

常用的原理图、PCB 设计工具有 Protel、PADS、OrCAD、Candence、Mentor 等。

3. 电路板调试

在拿到印制板厂家生产好的 PCB 板之后，首先需要进行裸板测试，确保电路板上没有短路、断路等情况，之后需要进行部分测试，最后进行总体测试。电路板调试的流程图如图 3-8 所示。

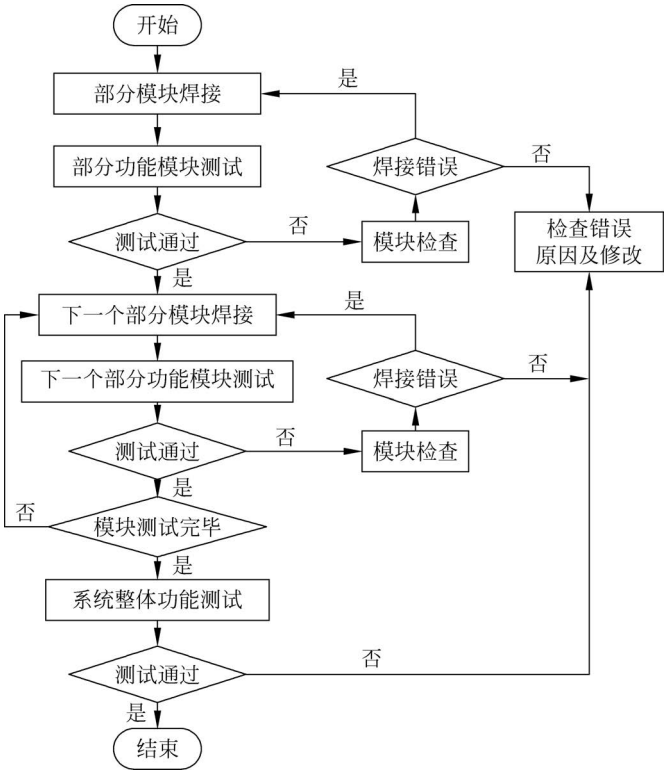


图 3-8　电路板调试流程图

在焊接元件过程中，应该先将所有元件归整、分类，并先焊接小元件，再焊接大元件。在对模块进行测试时，需要编写简单的硬件测试代码以验证模块是否正常工作，甚至还有可能要几个模块配合才能验证某个模块的功能是否正常。所有的验证通过之后则证明硬件的设计没有问题，之后就需要进行应用程序的开发，包括操作系统的移植、协议栈的实现、应用层程序的编写等工作。

▶ 3.2.5　常见传感器节点

1. Mica 系列节点

Mica 系列节点是美国加州大学伯克利分校研制的传感器网络演示平台，其软硬件设计都

是公开的，已经成为传感器网络的主要研究平台。Mica 系列节点包括 Renee、Mica、Mica2、Mica2Dot 和 MicaZ。TinyOS 是这些节点常用的操作系统。

表 3-6 给出了 Mica 系列节点的技术及性能指标。

表 3-6　Mica 系列节点性能指标

节点类型	Renee	Mica	Mica2	Mica2Dot	MicaZ
MCU 芯片类型	Atmega163	Atmega128			
UART 数量	1	2			
RF 芯片类型	TR1000		CC1000		CC2420
Flash 芯片类型	241.C256	AT45DB041B			
其他接口	DIO	DIO，I^2C	DIO，I^2C	DIO	DIO，FC
电源类型	AA	AA	AA	Lithium	AA
节点发布时间	1999	2001	2002	2002	2003

由表 3-6 可以看出，Mica 系列节点主要使用 Atmel 公司的处理器；Renee、Mica 节点使用 TR1000 无线通信芯片，Mica2、Mica2Dot 节点采用 CC1000 无线通信芯片，MicaZ 节点采用 CC2420 ZigBee 芯片。三种无线通信芯片的性能对比如表 3-7 所示。

表 3-7　通信芯片的性能指标

通信芯片类型	TR1000	CC1000	CC2420
载波技术	OOK/ASK	FSK	QPSK
线波频段/MHz	916	300～1000	2400
数据传输速率/($kb \cdot s^{-1}$)	OOK 方式：30 ASK 方式：115	76.8	250
接收最高灵敏度/dB	－105	－110	－95
信道数量	单信道	433MHz：3 个 868MHz：3 个 915MHz：43 个	16 个
通信距离/m	100～300	500～1000	60～150

从成本和功耗角度考虑，TR1000 和 CC1000 芯片是理想的选择，CC1000 灵敏度高且传播距离远，TR1000 功耗较低，CC2420 是较早支持 ZigBee 通信技术的通信芯片。

2. Telos 系列节点

Telos 系列节点是美国加州大学伯克利分校研究的成果，是针对 Mica 系列节点功耗较大而设计的低功耗产品。作为一个低功耗、可编程、无线传输的传感器网络硬件平台，Telos 节点具有两个基本模块，一是处理器和无线通信平台，二是传感器平台，两者之间通过标准接口连接。处理器和无线通信平台采用待机时功耗较低的 MSP430 处理器和 CC2420 无线收发芯片，Telos 系列节点使用两节 5 号干电池供电，待机时功耗为 2μW，工作时为 0.5mW，发送无线信号时为 45mW。从待机模式到工作模式转换时间为 270ns，最快为 6μs。当采用网格型网络拓扑结构，工作模式和待机模式的占空比采用不足 1%的设定，且与网络交换一次同步信号的情况下，最长可以工作 945 天。Telos 室外最长的传输距离达 100m，室内直线传输可达 50m。Telos 具有 A/D 转换器、D/A 转换器、UART、SM 等外围接口，具有强大的可扩展性。Telos 完全兼容 TinyOS。

3. BT 节点

BT 节点是一种多功能自主的无线通信和计算平台。它包括一个 Atmel ATmega 128L 处理器和随机访问内存及 128KB 闪存。与其他节点不同的是，BT 节点采用蓝牙技术，由工作

在433～915MHz的Chipon CC1000芯片构成，蓝牙射频和CC1000既可以单独工作，也可以同时工作，节点使用两节AA电池或3.8～5V外部电源供电。

4. Sun SPOT节点

Sun公司推出了一种新型的无线传感器网络设备Sun SPOT(Small Programmable Object Technology)。它采用32位的高性能ARM 920T处理器及支持ZigBee的CC2420无线通信芯片，并开发出Squawk Java虚拟机，可以使用Java语言搭建无线传感器网络。

处理器采用一款32位低功耗ARM 920T微处理器，相对于其他通用的微处理器，它具有更加丰富的资源和极低的功耗，不仅支持32位ARM指令集和16位Thumb指令集，拥有5级流水线和单一的32位AMBA总线接口，且含有MMU可以支持Windows CE、Linux等操作系统，具有统一的数据cache和指令cache。

Sun SPOT节点配有比较常见的传感器，如温度传感器、光强传感器和加速度传感器，节点可以利用输入/输出接口对传感器的功能进行扩展，节点提供了20个引脚来完成输入/输出工作。

在电源方面，节点内集成了一个3.7V、750mA的可充电锂电池，该电池拥有自保护机制，用于防止过度充放电、电压过载等异常情况，电池可以通过mini-USB口与计算机连接或通过外部5V电源进行充电。节点在深度睡眠的情况下可以运行909天，在全负荷运行的情况下最长可运行7小时。

5. Gain系列节点

Gain系列节点是中国科学院计算所开发的节点，是国内第一款自主开发的无线传感器网络节点。

Gain系列第一版节点的处理器采用中国科学院计算机所自行开发的处理器，该处理器采用哈佛总线结构，兼容AVR指令集。Gain系列的最新节点GAINSJ节点采用JENNIC SoC芯片JN5121，该芯片将处理器和射频芯片集成在一起，且兼容IEEE 802.15.4标准和ZigBee规范的协议栈，可以实现多种网络拓扑结构。节点休眠模式时的工作电流小于14mA，发送模式时的工作电流小于50mA，接收模式时的工作电流小于45mA，节点与PC采用RS232相连。

3.3 WSN的操作系统

▶ 3.3.1 WSN的操作系统概述

TinyOS是一个开源的嵌入式操作系统，它是由加州大学伯利克分校开发的，主要应用于无线传感器网络方面。它是一种基于组件(Component-Based)的架构，能够快速实现各种应用。TinyOS程序采用的是模块化设计，程序核心往往都很小。一般来说，核心代码和数据大概在400B左右，能够突破传感器存储资源少的限制，使得TinyOS可以有效地运行在无线传感器网络节点上，并负责执行相应的管理工作。

TinyOS本身提供了一系列的组件，可以很方便地编制程序，用来获取和处理传感器的数据，并通过无线方式来传输信息。可以把TinyOS看成一个与传感器进行交互的API，它们之间能实现各种通信。

在构建无线传感器网络时，TinyOS通过一个基地控制台即网关汇聚节点，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。

TinyOS的主要特点如下。

(1) 采用基于组件的体系结构,这种体系结构已经被广泛应用在嵌入式操作系统中。组件就是对软件、硬件进行功能抽象。整个系统由组件构成,通过组件提高软件的重用度和兼容性,程序员只关心组件的功能和自己的业务逻辑,而不必关心组件的具体实现,从而提高编程效率。

在TinyOS这种体系结构中,操作系统用组件实现各种功能,只包含必要的组件,提高了操作系统的紧凑性,减少了代码量和占用的存储资源。通过采用基于组件的体系结构,系统提供一个适用于传感器网络开发应用的编程框架,在这个框架内将用户设计的一些组件和操作系统组件连接起来,构成整个应用程序。

(2) 采用事件驱动机制,能够适用于节点众多、并发操作频繁发生的无线传感器网络。当事件对应的硬件中断发生时,系统能够快速地调用相关的事件处理程序,迅速响应外部事件,并且执行相应的操作处理任务。事件驱动机制可以使CPU在事件产生时迅速执行相关任务,并在处理完毕后进入休眠状态,有效提高了CPU的使用率,节省了能量。

(3) 采用轻量级线程技术和基于先进先出(First In First Out,FIFO)的任务队列调度方法。轻线程主要是针对节点并发操作可能比较频繁,且线程比较短,传统的进程/线程调度无法满足的问题提出的,因为使用传统调度算法会在无效的进程互换过程中产生大量能耗。

由于传感器节点的硬件资源有限,而且短流程的并发任务可能频繁执行,所以传统的进程或线程调度无法应用于传感器网络的操作系统。轻量级线程技术和基于FIFO的任务队列调度方法,能够使短流程的并发任务共享堆栈存储空间,并且快速地进行切换,从而使TinyOS适用于并发任务频繁发生的传感器网络应用。当任务队列为空时,CPU进入休眠状态,外围器件处于工作状态,任何外部中断都能唤醒CPU,这样可以节省能量。

(4) 采用基于事件驱动模式的主动消息通信方式,这种方式已经广泛用于分布式并行计算。主动消息是并行计算机中的概念。在发送消息的同时传送处理这个消息的相应处理函数和处理数据,接收方得到消息后可立即进行处理,从而减少通信量。由于传感器网络的规模可能非常大,导致通信的并行程度很高,传统的通信方式无法适应这样的环境。TinyOS的系统组件可以快速地响应主动消息通信方式传来的驱动事件,有效提高CPU的使用率。

TinyOS是一种面向传感器网络的新型操作系统,它最初是用汇编和C语言编写的,但C语言不能有效、方便地支持面向传感器网络的应用和操作系统的开发。科研人员对C语言进行了一定的扩展,提出了支持组件化编程的nesC语言,把组件化/模块化思想和基于事件驱动的执行模型结合起来。TinyOS操作系统、库程序和应用服务程序均是用nesC语言编写的。

▶ 3.3.2　nesC语言

nesC是对C的扩展,它基于体现TinyOS的结构化概念和执行模型而设计。TinyOS是为传感器网络节点而设计的一个事件驱动的操作系统,传感器网络节点拥有非常有限的资源(举例来说,8KB的程序存储器,512B的随机存取储存器)。TinyOS用nesC重新编写。

1. 简介

(1) 结构和内容的分离。程序由组件构成,它们装配在一起("配线")构成完整程序。组件定义两类域,一类用于它们的描述(包含它们的接口请求名称),另一类用于它们的补充。

组件内部存在作业形式的协作。控制线程可以通过它的接口进入一个组件。这些线程产生于一件作业或硬件中断。

(2) 根据接口的设置说明组件功能。接口可以由组件提供或使用。被提供的接口表现它为使用者提供的功能,被使用的接口表现使用者完成它的作业所需要的功能。

(3) 接口有双向性。它们叙述一组接口供给者(指令)提供的函数和一组被接口的使用者(事件)实现的函数。这允许一个单一的接口能够表现组件之间复杂的交互作用(如当某一事件在一个回调之前发生时,对一些事件的兴趣登记),这是危险的,因为 TinyOS 中所有的长指令(如发送包)是非中断的;它们的完成由一个事件(发送完成)标志。通过叙述接口,一个组件不能调用发送指令,除非它提供 sendDone 事件的实现。通常指令向下调用,例如,从应用组件到那些比较靠近硬件的调用,而事件则向上调用。特定的原始事件与硬件中断是关联的(这种关联是由系统决定的,因此在本书中不做进一步描述)。

(4) 组件通过接口彼此静态地相连。这可以增加运行时效率,支持鲁棒性设计,而且允许更好的程序静态分析。

(5) nesC 基于由编译器生成完整程序代码的需求设计。这考虑到较好的代码重用和分析。这方面的一个例子是 nesC 的编译-时间数据竞争监视器。

(6) nesC 的协作模型基于一旦开始直至完成作业,并且中断源可以彼此打断作业。nesC 编译器标记由中断源引起的潜在的数据竞争。

2. 使用环境编辑

nesC 主要用在 TinyOS 中,TinyOS 也是由 nesC 编写完成的。TinyOS 操作系统就是为用户提供一个良好的用户接口。基于以上分析,研发人员在无线传感器节点处理能力和存储能力有限的情况下设计一种新型的嵌入式系统 TinyOS,具有更强的网络处理和资源收集能力。为满足无线传感器网络的要求,研究人员在 TinyOS 中引入 4 种技术:轻线程、主动消息、事件驱动和组件化编程。轻线程主要是针对节点并发操作可能比较频繁,且线程比较短,传统的进程/线程调度无法满足(使用传统调度算法会产生大量能量用在无效的进程互换过程中)的问题提出的。

3. 主要特性描述

由于传感器网络的自身特点,面向其的开发语言也有其相应的特点。主动消息是并行计算机中的概念。在发送消息的同时传送处理这个消息的相应处理函数 ID 和处理数据,接收方得到消息后可立即进行处理,从而减少通信量。整个系统的运行是因为事件驱动而运行的,没有事件发生时,微处理器进入睡眠状态,从而可以达到节能的目的。组件就是对软硬件进行功能抽象。整个系统是由组件构成的,通过组件提高软件重用度和兼容性,程序员只关心组件的功能和自己的业务逻辑,而不必关心组件的具体实现,从而提高编程效率。

▶ 3.3.3 TinyOS 组件模型

TinyOS 包含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组件组合到各层配件文件中,并在面向应用程序的顶层配件文件中完成应用的整体装配。

TinyOS 的组件有 4 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(Data Frame),一组简单的任务。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，每个模块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。如图 3-9 所示显示了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，下层组件向上层组件发信号通知事件的发生，最底层的组件直接和硬件互通。

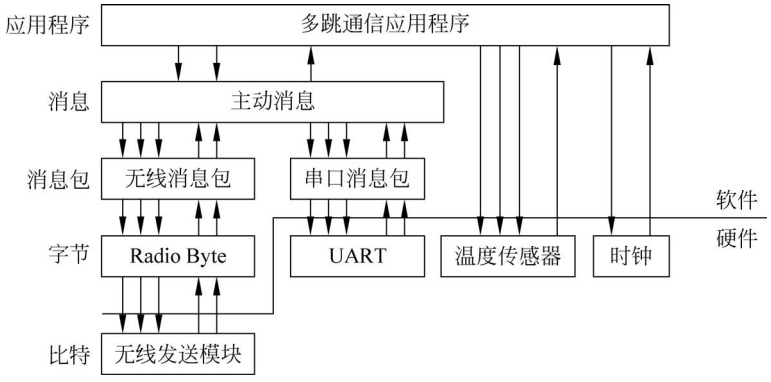


图 3-9　支持多跳无线通信的传感器应用程序的组件结构

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所需要的存储器空间。帧是一种特殊的符合 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 不提供动态的存储保护，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，帧的预分配可以防止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事件的开销，因为变量的位置在编译时就确定了，而不用通过指针动态地访问其状态。

在 TinyOS 中，命令是对下层组件的非阻塞请求。典型情况下，命令将请求的参数存储到本地的帧中，并为后期的执行有条件地产生一个任务（也称为轻量级线程）。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的发生。命令必须通过返回值为其调用者提供反馈信息，如缓冲区溢出返回失败等。

事件处理程序被激活，就可以直接或间接地去处理硬件事件。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。最底层的组件拥有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、定时器事件或者计数器事件。事件的处理程序可以存储消息到其所在帧中，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。硬件事件可以触发一连串的处理，其执行的方向，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。

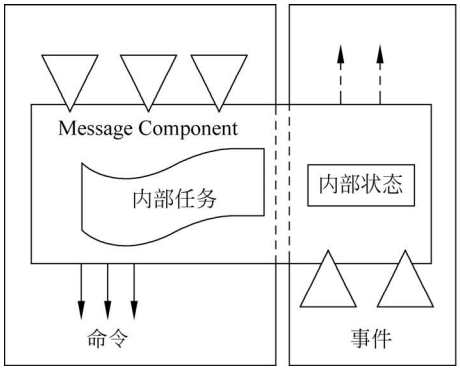
任务是完成 TinyOS 应用主要工作的轻量级线程。任务具有原子性，一旦运行就要运行至完成，不能被其他任务打断。但任务的执行可以被硬件中断产生的事件打断。任务可以调用下层命令，可以向上层发信号通知事件发生，也可以在组件内部调度其他任务。任务执行的原子特性，简化了 TinyOS 的调度设计，使得 TinyOS 仅分配一个任务堆栈就可以保存任务执行中的临时数据。该堆栈仅由当前执行的任务占有。这样的设计对于存储空间受限的系统来说是高效的。任务在每个组件中模拟了并发性，因为任务相对于事件而言是异步执行的。然而，任务不能阻塞，也不能空转等待，否则将会阻止其他组件的运行。

1. TinyOS 的组件类型

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、高层次的软件组件。硬件抽象组件将物理硬件映射到 TinyOS 组件模型。RFM(射频组件)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状态，如收发器处于发送模式还是接收模式、当前数据传输速率等。RFM 处理硬件中断并根据操作模式将其转换为接收(RX)比特事件或发送(TX)比特事件。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(如单独的 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的行为。这种组件的一个例子就是 Radio Byte 组件。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。从更高的层次上看，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，处理相同粒度的数据，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。这种类型组件的一个例子是如图 3-10 所示的主动消息处理模块。它履行在传输前填充包缓存区以及将收到的消息分发给相应任务的功能。执行鉴于数据或数据集合计算的组件也属于这一类型。




```
module TimerM {
    provides interface Timer[uint8_t id];
    provides interface StdControl;
    uses{
            interface Leds;
            interface Clock;
            interface PowerManagement;
    }
}...
```

图 3-10　主动消息处理模块

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。从软件移到硬件对于传感器网络来说是特别重要的，因为在传感器网络中，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。

3. 组件示例

图 3-10 是一个典型的组件，包含一个内部帧、事件处理程序句柄、命令和用于消息处理组件的任务。类似于大多数组件，它提供了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到达时，向相关组件发消息。为了完成这一功能，消息组件向完成数据包处理的下层组件发送命令并且处理两种类型的事件，其中一种表明传输完毕，另一种则表明已经收到一条消息。

组件描述了其提供的资源及其所要求的资源，将这些组件连接到一起就比较简单了。程序员要做的就是使一个组件所需要的事件和命令的特征与另一个组件所提供的事件和命令的特征相匹配。组件之间的通信采用函数调用的形式，这种方式系统开销小，能提供编译期的类型检查。

4. 组件组合

为了支持 TinyOS 的模块化特性，TinyOS 工作小组开发了一整套工具用于帮助开发者将组件连接起来。

在 TinyOS 中，组件在编译时被连接在一起，消除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。

在编译期间，为了创建把组件连接在一起的逻辑关系，会预先处理配件文件。这是由 nesC 编译器自动完成的。例如，单个事件可以被多个组件处理。编译期间可以自动生成代码，完成将事件通知到相关组件的事件处理函数的功能。nesC 编译器的输出是一个标准 C 文件，包含应用程序中的所有组件，也包含所有必需的连接信息。

▶ 3.3.4　TinyOS 通信模型

TinyOS 中的消息通信遵循主动消息通信模型，它是一个简单的、可扩展的、面向消息通信(Messaged-based Communication)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(Application-Layer)的处理器(handler)。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进行处理。应用层的处理器一般完成消息数据的解包操作、计算处理或发送响应消息等工作。在这种情况下，网络就像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碰到的缓冲区处理方面的困难情况。为了避免网络拥塞，还需要消息处理器能够实现异步执行机制。

尽管主动消息起源于并行和分布式计算领域，但其基本思想适合传感器网络的需求。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使用忙等(busy-waiting)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，并减少传感器节点的能耗。

1. 主动消息的设计实现

在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节点的计算和通信重叠，让软件层的通信原语能够与无线传感器节点的硬件能力匹配，充分节省无线传感器节点的有限存储空间。可以把主动消息通信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于传感器网络的需求，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需求。如果把主动消息通信实现为一个 TinyOS 的系统组件，则可以屏蔽下层各种不同的通信硬件，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，如可靠传输的组件、加密传输的组件等。这样上层应用程序可以根据具体需求，选择合适的通信组件。在传感器网络中，由于应用千差万别和硬件功能有限，TinyOS 不可能提供功能复杂的通信组件，而只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。

2. 主动消息的缓存管理机制

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有显著影响。当数据通过网络到达传感器节点时，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。

如果传感器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 为了解决这个问题，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，用于接收下一个将要到来的消息。在 TinyOS 中，各个应用程序之间的执行是不能抢占的，所以不会出现多个未使用的消息缓存发生冲突，这样 TinyOS 的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。

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(Private Frame)上静态分配额外的空间以保存消息。实际上，在 TinyOS 中，所有的数据分配都是在编译时确定的。

3. 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确认消息包，这样比在应用层生成确认消息包节省开销，反馈时间短。为了进一步节省开销，TinyOS 仅发送一个特殊的立即数序列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要求重新发送消息。从总体上看，这种简单的显式确认通信机制适合传感器网络的有限资源，是一种有效的通信手段。

▶ 3.3.5 TinyOS 事件驱动机制

为了满足无线传感器网络需要的高水平的运行效率，TinyOS 使用基于事件的执行方式。事件模块允许高效的并发处理运行在一个较小的空间内。相比之下，基于线程的操作系统则需要为每个上下文切换预先分配堆栈空间。此外，线程系统上下文切换的开销明显高于基于事件的系统。

为了高效地利用 CPU，基于事件的操作系统将产生低功耗的操作。限制能量消耗的关键因素是如何识别何时没有重要的工作去做而进入极低功耗的状态。基于事件的操作系统强迫应用使用完毕 CPU 时隐式声明。在 TinyOS 中当事件被触发后，与发出信号的事件关联的所有任务将被迅速处理。当该事件以及所有关联任务被处理完毕时，未被使用的 CPU 循环被置于睡眠状态而不是积极寻找下一个活跃的事件。TinyOS 这种事件驱动方式使得系统高效地使用 CPU 资源，保证了能量的高效利用。TinyOS 这种事件驱动操作系统，当一个任务完成后，就可以使其触发一个事件，然后 TinyOS 就会自动地调用相应的处理函数。

事件驱动分为硬件事件驱动和软件事件驱动。硬件事件驱动是一个硬件发出中断，然后

进入中断处理函数；而软件驱动则是通过 signal 关键字触发一个事件。这里所说的软件驱动是相对于硬件驱动而言的，主要用于在特定的操作完成后，系统通知相应程序做一些适当的处理。以 Blink 程序为例阐述硬件事件处理机制。Blink 程序中，定时器每隔 1000ms 产生一个硬件时钟中断。在基于 ATMega 128L 的节点中，时钟中断是 15 号中断。如图 3-11 所示，通过调用 BlinkM. StdControl. start()开启定时器服务。

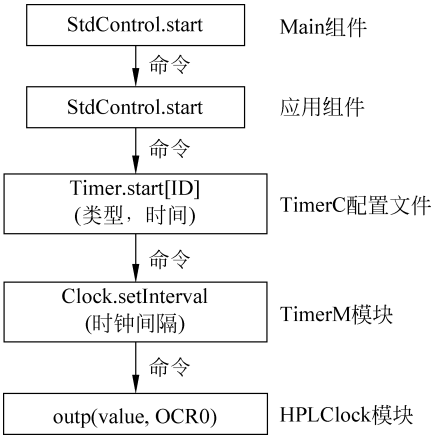


图 3-11　定时器服务启动流程

一个时钟中断向量表是处理器处理中断事件的函数调度表格，它的位置和格式与处理器设计相关。有的处理器规定中断向量表直接存放中断处理函数的地址，由处理器产生跳转指令进入处理地点，如中断向量存放 0x3456，处理器在发生中断时，组织一条中断调用指令执行 0x3456 处的代码；还有一些则是为每个中断在表中提供一定的地址空间，产生中断时系统直接跳转到中断向量的位置执行。后一种情况直接在中断向量处存放中断处理代码，不用处理器组织跳转指令。不过一般预留给中断向量的空间有限，如果处理函数比较复杂，一般都会在中断向量的位置保存一条跳转指令。ATMega 128L 处理器的中断向量的组织使用的是后一种处理方式。

中断向量表是在编译连接时根据库函数的定义连接的。连接后的中断向量表如图 3-12 所示。0 号中断是初始化(reset)中断，1～43 号中断根据各个处理器的不同有可能不同，ATMEGA 128L 中的 Timer 对应 15 号中断。于是在地址 0x3c 处的指令就是跳转到中断入口处，也就是 vector 15 处。而其他中断没有给定处理函数，所以就跳到 0xc6 处，也就是 ad interrupt 处理程序。

由此可知，实际上 TinyOS 把定时器安装到中断号为 15 的中断向量表中了。当定时器中断发生时，就会执行地址 0x3c 处的指令，jmp 0x318 处去执行，现文件 HPLClock 也就是程序中的 vector 15 处。vector 15 是在 Clock 接口中实现的。定时器发生中断的响应过程如图 3-13 所示。

```
00000000<vectors>:
0:0c 94 46 00 jmp 0x8c
4:0c 94 63 00 jmp 0xc6
8:0c 94 63 00 jmp 0xc6
⋮
3c:0c 94 8c 01 jmp 0x318//_vector_15对应的程序入口
40:0c 94 63 00 jmp 0xc6
⋮
000000c6<_bad_interrupt>:
c6:0c 94 00 00 jmp 0x0
```

图 3-12　ATMEGA 128L 中断向量表

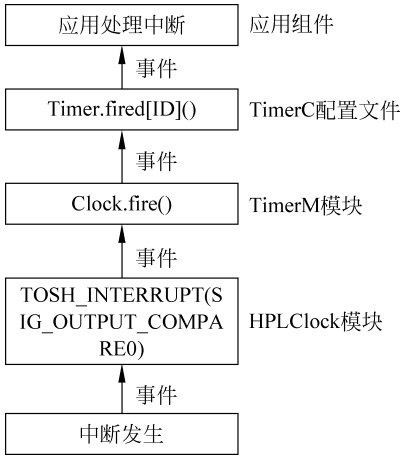


图 3-13　定时器服务响应中断流程

▶ 3.3.6 调度策略

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。

事件驱动的 TinyOS 采用两级调度：任务和硬件事件处理句柄（Hardware Event Handlers）。任务是一些可以被抢占的函数，一旦被调度，任务运行完成前彼此之间不能相互抢占。硬件事件处理句柄被执行去响应硬件中断，可以抢占任务的运行或者其他硬件事件处理句柄。TinyOS 的任务调度队列只是采用简单的 FIFO 算法。任务事件的调度过程如图 3-14 所示。TinyOS 的任务队列如果为空，则进入极低功耗的 SLEEP 模式。当被事件触发后，在 TinyOS 中发出信号的事件关联的所有任务被迅速处理。当这个事件和所有任务被处理完成后，未被使用的 CPU 循环被置于睡眠状态而不是积极寻找下一个活跃的事件。

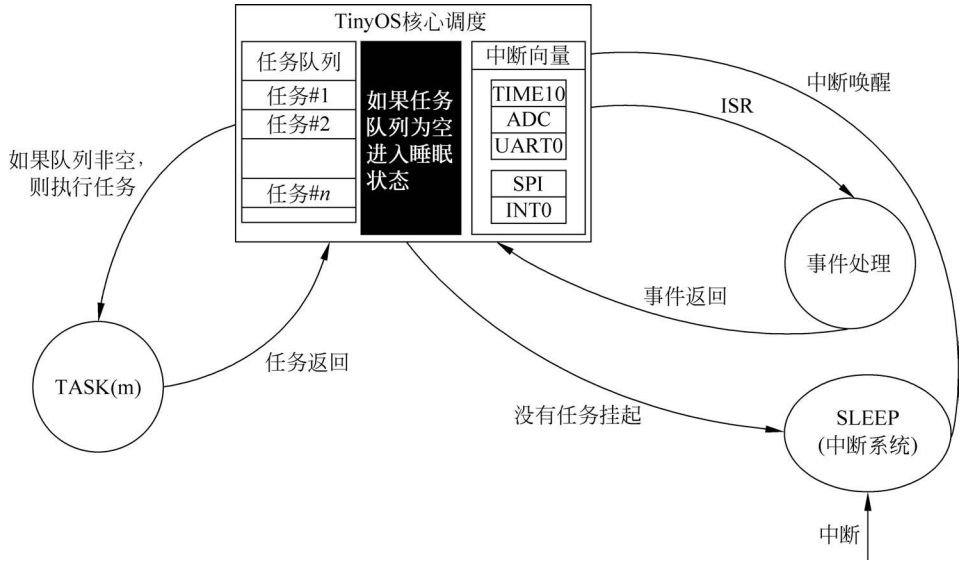
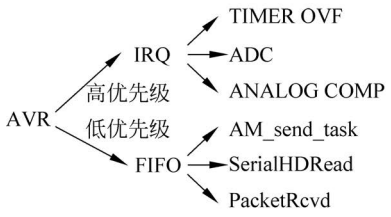


图 3-14 TinyOS 任务事件调度过程

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），如图 3-15 所示。任务之间不允许相互抢占；而硬件事件句柄，即中断处理线程可以抢占用户的任务和低优先级的中断处理线程，保证硬件中断快速响应。TinyOS 的任务队列如果为空，则让处理器进入极低功耗的 SLEEP 模式。但是保留外围设备的运行，以至于它们中的任何一个可以唤醒系统。部分调度程序源代码如图 3-15 所示，其中，TOSH_run_next_task()函数判断队列是否为空，如果是则返回 0，系统进入睡眠模式，否则做出队列操作并执行队列该项所指向的任务并返回 1。一旦任务队列为空，另一个任务能被调度的唯一条件是事件触发的结果；因而不需要唤醒调度程序直到硬件事件的触发活动。

TinyOS 的调度策略具有能量意识。




```
bool TOSH_run_next_task(){
  if(TOSH_sched_full==
TOSH_sched_free){
    return 0;
}
else{
  移除队列头的任务;
  执行该任务;
  return l;
     }
                              }
void TOSH_run_task(){
  while(TOSH_run_next_task())
    ;
  TOSH_sleep();
  TOSH_wait();
                    }
```

图 3-15　TinyOS 的调度结构以及部分调度程序源代码

▶ 3.3.7　能量管理机制

无线传感器网络节点运行在人无法接近甚至危险的远程环境中，加上电源能量有限，所以设计有效的策略来减少能量消耗、延长网络生存时间一直是研究的热点和难点。无线传感器网络的能量问题需要考虑无线传感器网络的特性，并将多种性能指标结合起来考虑。这是一个涉及软硬件，涉及多层通信协议的复杂问题。

从节点操作系统这一层面上看，TinyOS 采用相互关联的三部分进行能量管理。第一，每个设备都可以通过调用自身的 StdControl. stop 命令停止该设备；负责管理外围硬件设备的组件将切换该设备到低功耗状态。第二，TinyOS 通过提供 HPLPowerManagement 组件通过检测处理器的 I/O 引脚和控制寄存器识别当前硬件的状态将处理器转入相应的低功耗模式。第三，TinyOS 的定时器服务可以工作在大多数处理器极低功耗的省电模式下。

在 TinyOS 中，采用的是简单的 FIFO 队列，不存在优先级的概念。事件驱动的 TinyOS，如果任务队列为空，则进入睡眠态，直到有事件唤醒，才去处理事件以及与事件相关的所有任务，然后再次进入睡眠态。因而这种事件驱动的驱动系统，保证节点大多数时期都处在极低功耗的睡眠状态，有效地节约了系统的能量消耗，延长了传感器网络的生命周期。而基于多任务的系统，实际上不考虑低功耗的应用。

由于多任务的系统需要进行任务切换或者中断服务与任务间的切换。而每次切换就是保存正在运行的任务的当前状态，即 CPU 寄存器中的全部内容，这些内容保存在运行任务的堆栈内。入栈工作完成后，就把下一个将要运行的任务的当前状况从任务的堆栈中重新装入 CPU 的寄存器中，并开始下一个任务的运行。

在事件驱动的 TinyOS 中，由于对任务的特殊语义定义运行-完成(run to completion)。任务之间是不能切换的，所以任务的堆栈是共享的，并且任务堆栈总是当前正在运行的任务在使用。从运行空间方面看，多任务系统需要为每个上下文切换预先分配空间，而事件驱动的执行模块则可以运行于很小的空间中。因此多任务系统的上下文开销要明显高于事件系统，TinyOS 更好地减小了系统对 ROM 的需求量，满足了节点内存资源有限的限制。

▶ 3.3.8　LED 灯闪烁实验分析

1. 原理

无线传感器网络实验平台设备上有 D13、D14、D15、D16 这 4 个 LED 灯，其中，D13 和 D15

主要用于程序调试，可以根据具体功能进行相应的更改。通过原理图可知，当 CC2530 与 LED 相接的数字 I/O 输出高电平时 LED 点亮，输出低电平时 LED 熄灭，输出交替电平时 LED 闪烁。在本实验中，系统启动后，D13 和 D15 轮流点亮，点亮和变暗的间隔用 for 循环延时实现。LED 控制电路图如图 3-16 所示。

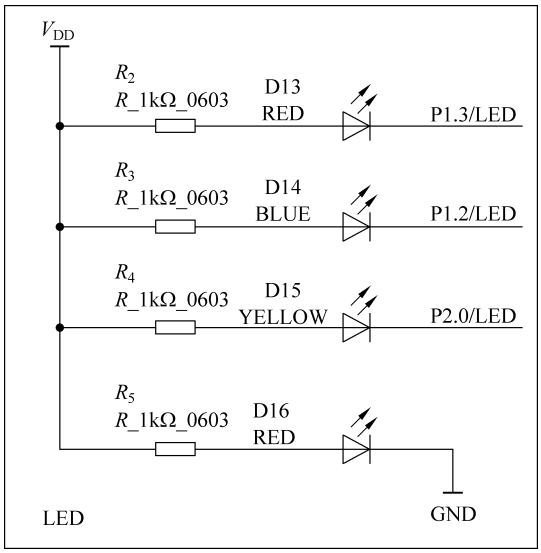


图 3-16　LED 控制电路图

2. 相关代码

BlinkM. nc 文件：

```
/ ************************************************
 * FUNCTION NAME : BlinkM.nc
 * FUNCTION DESCRIPTION : LED 灯闪烁
 * FUCNTION DATE :2011/11/29
 * FUNCTION AUTHOR: EMDOOR
 ** /
module BlinkM {
    uses interface Leds;
    uses interface Boot;
}
implementation {
    task void DemoLed(){
        int i,j;
        while(1) {
            for(i = 0;i < 1000;i++)
                for(j = 0;j < 500;j++);
            call Leds.BlueLedOn();          //D14 LED 亮
            call Leds.RedLedOff();          //D15 LED 亮
            for(i = 0;i < 1000;i++)
                for(j = 0;j < 500;j++);
            call Leds.BlueLedOff();         //D14 LED 灭
            call Leds.RedLedOn();           //D15 LED 亮
        }
    }
    / ** 启动事件处理函数，在 LED.nc 已经关联到 MainC.Boot 接口
```

```
        系统启动后会调用此函数
    */
    event void Boot.booted() {
        post DemoLed();
    }
}
```

BlinkC.nc 文件：

```
/**************************************************
* FUNCTION NAME : BlinkC.nc
* FUNCTION DESCRIPTION : LED灯闪烁
* FUCNTION DATE :2010/10/14
* FUNCTION AUTHOR: EMDOOR
**/
configuration BlinkC
{
}
implementation
{
    components BlinkM;                          //LED模块程序,用于实现LED代码
    components LedsC;
    components MainC;                           //TinyOS2主模块,这里用于关联系统启动
    /* LED模块程序中控制LED的I/O与TinyOS提供的接口相关联 */
    BlinkM.Leds -> LedsC;
    /* LED模块程序的Boot接口与系统Boot接口
     * 关联,这样系统启动时会调用LedM的Boot接口 */
    BlinkM.Boot -> MainC.Boot;
}
```

Makefile 文件：

```
COMPONENT=BlinkC
########################
PFLAGS += -DUSE_MODULE_LED
########################
include $(MAKERULES)
```

3. 查看实验结果

节点板上的 D14(LED灯)和 D15(LED灯)依次点亮、熄灭。

3.4　现代 WSN 典型实验平台

当今市场上无线传感器网络实验种类繁多，但功能基本相同。下面以武汉创维特 CVT-WSN-Ⅱ无线传感器网络教学实验系统为例介绍无线传感器网络典型实验平台的硬件组成。

▶ 3.4.1　硬件系统的组成

CVT-WSN-Ⅱ无线传感器综合教学实验系统包括 16 种传感器模块、5 种被控单元、无线射频模块。其中，传感器模块包括温度、温湿度、光照、人体感应、振动、可燃气体、酒精、压力、气象气体压力、超声波测距、三轴加速度、水流量、雨滴、霍尔、磁场等，被控单元包括 LED 矩阵、数码管、蜂鸣器、步进电机、直流电机，网关直接采用 TI 公司的无线单片机 CC2530 作为核心处理器，其硬件系统组成如图 3-17 和表 3-8 所示。

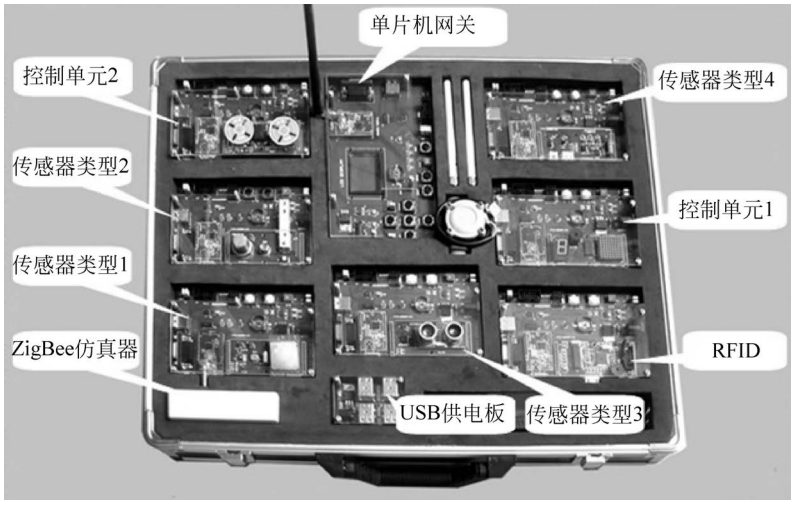


图 3-17　CVT-WSN-Ⅱ教学实验系统硬件组成

表 3-8　CVT-WSN-Ⅱ教学实验系统硬件组成

序号	名称	规格型号	数量	备　注
1	无线单片机显示网关板	CVT-WSNMCULCD	1	CC2530 处理器，128×64 点阵 LCD 屏 7 个功能按键，4 个 LED 灯
2	通用调试母板	CVT-WSN-EMK	7	3 个按键/1 个 AD/3 个 LED/1 个 UART，3 种供电模式
3	ZigBee 板	CVT-ZIGBEE	8	(1) 工作频段 2.4GHz，信道 16 个 (2) 信道带宽 5MHz (3) 符合 IEEE 802.15.4 标准 (4) 内置 IEEE 802.15.4 MAC 协议栈
4	传感器扩展板 1	CVT-WSNSENSOR1	1	温度/温湿度/光照/人体感应/振动
5	传感器扩展板 2	CVT-WSNSENSOR2	1	可燃气体/酒精/压力/气象气体压力
6	传感器扩展板 3	CVT-WSNSENSOR3	1	超声波测距/三轴加速度
7	传感器扩展板 4	CVT-WSNSENSOR4	1	水流量/雨滴/霍尔/磁场
8	指示扩展板	CVT-WSNSEG	1	LED/数码管/蜂鸣器指示
9	控制扩展板	CVT-WSNMOTOR	1	直流电机/步进电机/继电器控制
10	RFID 扩展板	CVT-WSNRFID	1	13.56MHz RFID
11	USB 供电板	CVT-WSNUSB	1	供电电源线一个，USB 线 8 根
12	ZigBee 通用仿真监视器	CC2000	1	下载调试排线一根 USB 2.0 下载调试线一根

▶ 3.4.2　硬件组件介绍

1. 无线单片机显示网关板

无线单片机显示网关板可以在实验中将无线传感器的相关设备信息传送到 PC 监控软件，也可以将 PC 监控软件的控制信息通过射频发送到控制单元，对被控单元进行相关的控制。

2. 通用调试母板

提供 ZigBee 板、传感器扩展板、指示扩展板、电机控制扩展板、RFID 扩展板的基板，也可以作为简易的网关板使用，或与 CC2000 仿真器一起作为 ZigBee 协议监视分析硬件平台。支

持三种供电方式，包括 USB 供电、外接电源供电以及纽扣电池供电，调试监视方式支持标准串口方式和 USB 串口方式。

3. ZigBee 板

采用 TI 公司的 CC2530 ZigBee 无线单片机，内含 128KB 的 Flash 存储器，ZigBee 协议栈采用 IEEE 802.15.4 标准。此板外形小巧，使用方便，供电后即可工作。支持外接天线或片式天线。

4. 传感器扩展板

传感器扩展板支持包括温度、温湿度、光照、人体感应、振动、可燃气体、酒精、压力、气象气体压力、超声波测距、三轴加速度、水流量、雨滴、霍尔、磁场传感器等，还可根据学校的要求进行定制。

为了使用方便，将传感器进行了分类，按功能或大小，将上面的传感器组合为 4 种传感器板，即传感器扩展板 1、传感器扩展板 2、传感器扩展板 3、传感器扩展板 4。下面对 4 个传感器板分别进行介绍。

(1) 传感器扩展板 1：包含温度传感器、温湿度传感器、光照传感器、人体感应传感器及振动传感器。

(2) 传感器扩展板 2：包含可燃气体传感器、酒精传感器、压力传感器及气象气体压力传感器。

(3) 传感器扩展板 3：包含超声波测距传感器、三轴加速度传感器。

(4) 传感器扩展板 4：包含水流量传感器、雨滴传感器、霍尔传感器、磁场传感器。

(5) 指示扩展板：指示扩展板包含 LED 指示、七段数码管指示、8×8 点阵 LED 指示、蜂鸣器指示功能。

(6) 控制扩展板：控制扩展板包含直流电机控制、步进电机控制、继电器控制。

(7) RFID 扩展板：RFID 扩展板支持非接触式 IC 卡读写，支持 ISO 14443 RFID 卡协议，频率 13.56MHz。

(8) ZigBee 仿真器：ZigBee 仿真器用于无线单片机 CC2530 的程序下载、调试，程序的在线烧写，协议抓包分析等，如图 3-18 所示。

图 3-18　ZigBee 仿真器

注：红色 Power 指示灯为电源指示；红色 NoTarget 指示灯为未找到目标板指示；绿色 Link 指示灯为找到目标板指示。

仿真器具有在线下载、调试、仿真等功能，其外形非常简单，具有一个 USB 接口、一个指示灯、一根仿真下载线。

(1) USB 接口：通过 USB 接口把仿真器与计算机连接起来。仿真器通过此接口与计算机通信，要在无线网络接点模块上实现下载、调试(Debug)、仿真等的通信都由此接口来实现。

(2) 指示灯：电源指示灯。

(3) 仿真线：这是一根下载、调试仿真线，通过它与无线节点模块或开发板连接。

▶ 3.4.3　传感器节点

传感器节点外观组成如图 3-19 和图 3-20 所示，主要包括一块底板(采集板)与一块 ZigBee

模块，根据需要可增加传感器扩展板。典型底板型号为 C51RF-WSN-DA100，传感器扩展板型号为 C51RF-WSN-DA300。

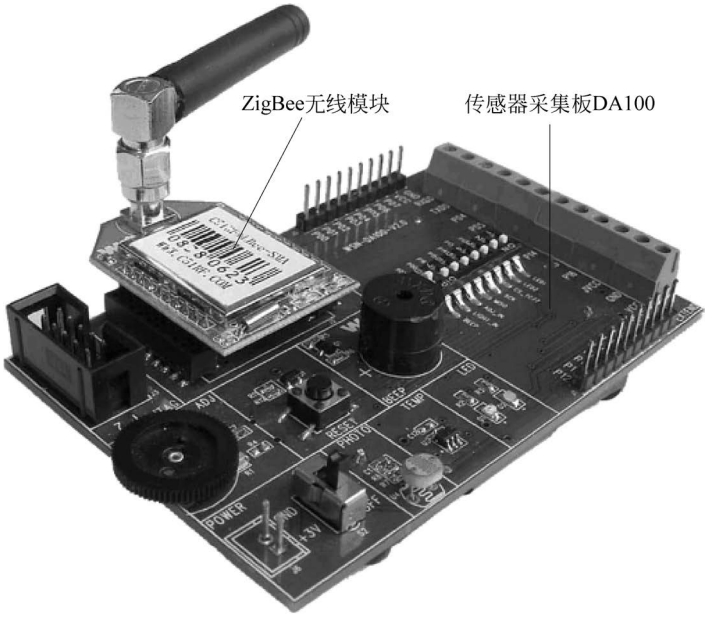


图 3-19　传感器节点

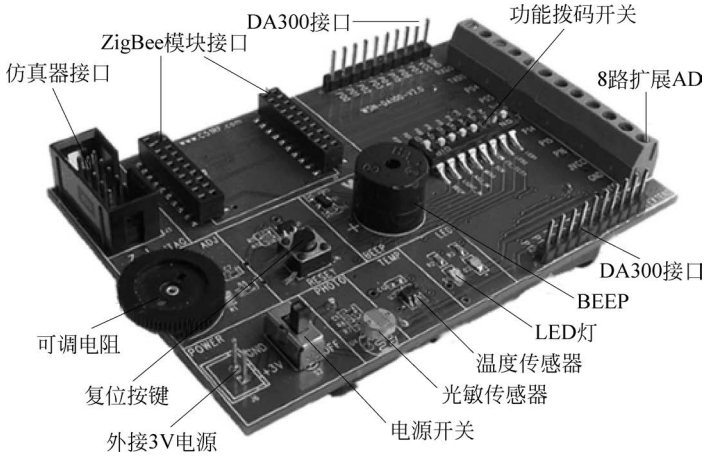


图 3-20　传感器底板（采集板）DA100

传感器节点在无线 ZigBee 传感器网络系统中既可用作普通传感器节点，也可用作无线 ZigBee 网络中的路由节点。传感器节点主要实现温度采集、光照度采集、BEEP（蜂鸣器）、LED 测试小灯、数据发送等功能。传感器在采集到温度值或者光照度值后，通过 ZigBee 模块内的 CC2530 单片机的 AD 将其转换为数字电压，通过 ZigBee 无线模块的射频部分将其发送给路由节点或者网关。可以通过 PC 软件对它进行直接访问，例如，可以通过 PC 使传感器节点完成 LED 小灯的测试。传感器采集板 DA100 包括温度传感器、光敏传感器、可调电阻、BEEP（蜂鸣器）、8 路可扩展 AD、仿真器接口、ZigBee 无线模块接口、DA300 接口、功能拨码开关等。

▶ 3.4.4　路由器节点

在传感器节点不能和网关直接通信的时候，路由器节点就起到了连接网关和传感器节点通信的目的。传感器节点硬件电路在无线 ZigBee 传感器网络系统既可用作普通传感器节点，也可用作无线 ZigBee 网络中的路由节点。当传感器节点用作路由节点时，硬件电路配置如图 3-19 所示。

3.5　ZigBee 硬件平台

TI 公司的 CC2530 是真正的系统级 SoC 芯片，适用于 2.4GHz IEEE 802.15.4，ZigBee 和 RF4CE 应用。CC2530 包括极好性能的一流的 RF 收发器，工业标准增强型 8051 MCU，系统中可编程的闪存(8KB RAM)具有不同的运行模式，使得它尤其适应超低功耗要求的系统，以及许多其他功能强大的特性，结合 TI 公司的业界领先的黄金单元 ZigBee 协议栈(Z-Stack™)，提供了一个强大和完整的 ZigBee 解决方案。

▶ 3.5.1　CC2530 芯片的特点

CC2530 是一个真正的用于 2.4GHz IEEE 802.15.4 与 ZigBee 应用的 SoC 解决方案。这种解决方案能够提高性能并满足以 ZigBee 为基础的 2.4GHz ISM 波段应用对低成本、低功耗的要求。它结合了一个高性能 2.4GHz DSSS(直接序列扩频)射频收发器核心和一颗工业级小巧、高效的 8051 控制器。

CC2530 芯片方框图如图 3-21 所示。内含模块大致可以分为三类：CPU 和内存相关的模块，外设、时钟和电源管理相关的模块，以及射频相关的模块。CC2530 在单个芯片上整合了 8051 兼容微控制器、ZigBee 射频(RF)前端、内存和 Flash 存储器等，还包含串行接口(UART)、模/数转换器(ADC)、多个定时器(Timer)、AES128 安全协处理器、看门狗定时器(Watchdog Timer)、32kHz 晶振的休眠模式定时器、上电复位电路(Power On Reset)、掉电检测电路(Brown Out Detection)以及 21 个可编程 I/O 口等外设接口单元。

CC2530 的主要特点如下。

- 高性能、低功耗、带程序预取功能的 8051 微控制器内核。
- 32KB/64KB/128KB/256KB 的系统可编程 Flash。
- 8KB 在所有模式都带记忆功能的 RAM。
- 2.4GHz IEEE 802.15.4 兼容 RF 收发器。
- 优秀的接收灵敏度和强大的抗干扰性。
- 精确的数字接收信号强度(RSSI)指示/链路质量指示(LQI)支持。
- 最高到 4.5dBm 的可编程输出功率。
- 集成 AES 安全协处理器，硬件支持的 CSMA/CA 功能。
- 具有 8 路输入和可配置分辨率的 12 位 ADC。
- 强大的 5 通道 DMA。
- IR 发生电路。
- 带有两个强大的支持几组协议的 UART。
- 一个符合 IEEE 802.15.4 规范的 MAC 定时器、一个常规的 16 位定时器和两个 8 位定时器。

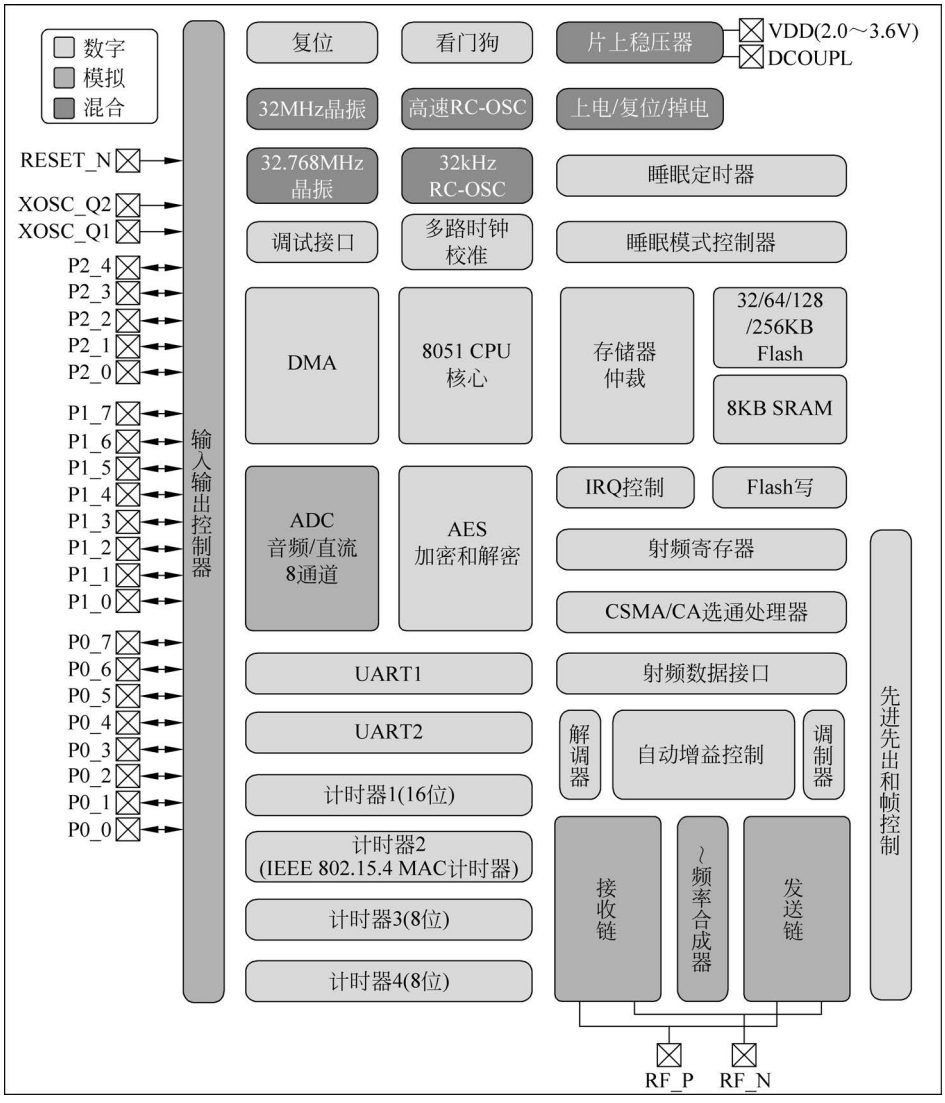


图 3-21　CC2530 芯片方框图

- 看门狗定时器，具有捕获功能的 32kHz 睡眠定时器。
- 较宽的电压工作范围(2.0～3.6V)。
- 具有电池监测和温度感测功能。
- 在休眠模式下仅 0.4μA 的电流损耗，外部中断或 RTC 能唤醒系统。
- 在待机模式下低于 1μA 的电流损耗，外部中断能唤醒系统。
- 调试接口支持，强大和灵活的开发工具。
- 仅需很少的外部元件。

▶ 3.5.2　CC2530 片上 8051 内核

CC2530 芯片使用的 8051 CPU 内核是一个单周期的 8051 兼容内核。它有三种不同的内存访问总线(SFR、DATA 和 CODE/XDATA)，单周期访问 SFR、DATA 和主 SRAM。它还包括一个调试接口和一个 18 位输入扩展中断单元。中断控制器总共提供了 18 个中断源，分

为6个中断组，每个与4个中断优先级之一相关。当设备从IDLE模式回到活动模式时，任一中断服务请求也能响应。一些中断还可以从睡眠模式唤醒设备。

内存仲裁器位于系统中心，因为它通过SFR总线把CPU、DMA控制器和物理存储器以及所有外设连接起来。内存仲裁器有4个内存访问点，每次访问可以映射到三个物理存储器之一：一个8KB SRAM、闪存存储器和XREG/SFR寄存器。它负责执行仲裁，并确定同时访问同一个物理存储器之间的顺序。

1. 增强型8051内核

增强型8051内核使用8051指令集。指令运行比标准的8051更快，因为：

- 每条指令一个时钟周期，而普通8051为每条指令12个时钟周期。
- 除去了被浪费掉的总线状态：因为一条指令周期是和可能的存储器获取对齐的，大部分单指令的执行时间为一个系统时钟周期。为了速度的提高，CC2530增强型内核还增加了两部分：另一个数据指针以及扩展的18个源的中断单元。
- CC2530内核的目标代码兼容标准8051微处理器。换句话说，CC2530的8051目标码与标准8051完全兼容，可以使用标准8051的汇编器和编译器进行软件开发，所有CC2530的8051指令在目标码和功能上与同类标准的8051产品完全等价。不管怎样，由于CC2530的8051内核使用不同于标准的指令时钟，且外设如定时器、串口等不同于标准的8051，因此在编程时与标准的8051代码略有不同。

2. 存储空间

CC2530包含一个DMA控制器，8KB静态RAM(SRAM)，32KB、64KB、128KB或256KB的片内提供在系统可编程的非易失性存储器(Flash)。

8051 CPU结构有4个不同的存储器空间。8051有独立的程序存储器和数据存储器空间。

(1) CODE程序存储器空间：一块只读程序存储器空间，地址空间为64KB。

(2) DATA数据存储器空间：一块8位的可读/可写的数据存储器空间，可通过单周期的CPU指令直接或间接存取。地址空间为256B，低128B可通过直接或间接寻址访问，而高128B只能通过间接寻址访问。

(3) XDATA数据存储器空间：一块16位的可读/可写的数据存储空间，通常访问需要四五个指令周期，地址空间为64KB。

(4) SFR特殊功能寄存器：一块可通过CPU的单周期指令直接存取的可读/可写寄存器空间。地址空间为128B，特殊功能寄存器可进行位寻址。

以上4块不同的存储空间构成了CC2530的存储器空间，可通过存储管理器来进行统一管理。

3. 特殊功能寄存器

特殊功能寄存器控制CC2530的8051内核以及外设的各种重要功能。大部分的CC2530特殊功能寄存器与标准8051特殊功能寄存器功能相同，小部分与标准8051的不同，不同的特殊功能寄存器主要用于控制外设以及射频收发功能。

▶ 3.5.3　CC2530主要特征外设

CC2530有21个数字I/O引脚，能被配置为通用数字I/O口或作为外设I/O信号连接到ADC、定时器或串口外设。

1. 输入/输出接口

CC2530 包括三组输入/输出(I/O)口,分别是 P0、P1、P2。其中,P0 和 P1 分别有 8 个引脚,P2 有 5 个引脚,共 21 个数字 I/O 引脚。这些引脚都可以用作通用的 I/O 端口,同时通过独立编程还可以作为特殊功能的输入/输出,通过软件设置还可以改变引脚的输入/输出硬件状态配置。

2. 直接存取控制器

中断方式解决了高速内核与低速外设之间的矛盾,从而提高了单片机的效率。但在中断方式中,为了保证可靠地进行数据传送,必须花费一定的时间,如重要信息的保护以及恢复等,而它们都是与输入/输出操作本身无关的操作。因此对于高速外设,采用中断模式就会感到吃力。为了提高数据的存取效率,CC2530 专门在内存与外设之间开辟了一条专用数据通道。这条数据通道在 DMA 控制器硬件的控制下,直接进行数据交换而不通过 8051 内核,不用 I/O 指令。

DMA 控制器可以把外设(如 ADC、射频收发器)的数据移到内存而不需要 CC2530 内核的干涉。这样,传输数据速度上限取决于存储器的速度。采用 DMA 方式传送时,由 DMA 控制器向 8051 内核发送 DMA 请求,内核响应 DMA 请求,这时数据输入/输出完全由 DMA 控制器指挥。

3. 定时器

CC2530 包含两个 16 位的定时器/计数器(Timer1 和 Timer2)和两个 8 位的定时器/计数器(Timer3 和 Timer4)。其中,Timer2 是主要用于 MAC 的定时器。

Timer1、Timer3、Timer4 为支持典型的如输入、捕获、输出比较与 PWM 功能的定时器/计数器。这些功能和标准的 8051 是差不多的。Timer2 主要用于 802.15.4 CSMA-CA 算法与 802.15.4 MAC 层的计时。如果定时器 2 与睡眠定时器一起使用,当系统进入低功耗模块时,定时器 2 将提供定时功能,使用睡眠定时器设置周期。

4. 14 位模/数转换器

CC2530 的 ADC 支持 14 位的模/数转换,这跟一般的单片机 8 位 ADC 不同,如图 3-22 所示。这个 ADC 包括一个参考电压发生器和 8 个独立可配置通道。转换结果可通过 DMA 写到存储器中,有多种操作模式。

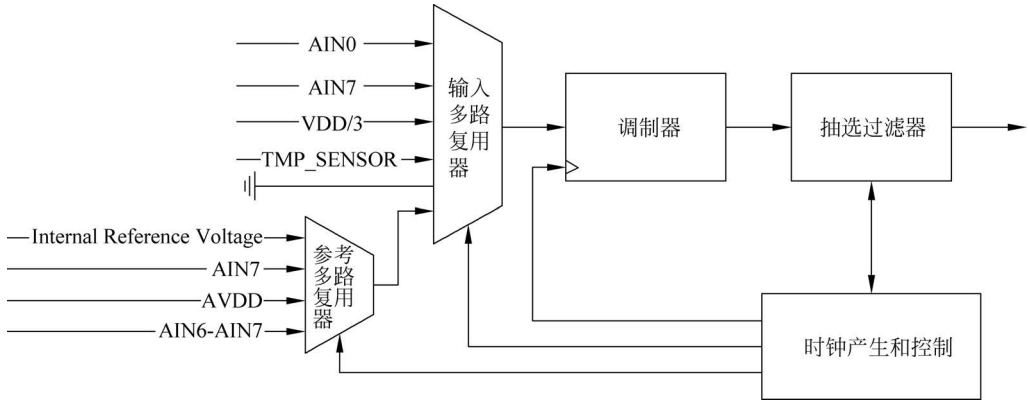


图 3-22　ADC 方框图

5. 串行通信接口

CC2530 有两个串行接口:USART0 和 USART1。可以独立操作在异步 UART 模式或

同步 SPI 模式。两个 USART 有相同的功能，对应分配到不同的 I/O 口，2 线制或 4 线制，硬件流控支持。

(1) UART 模式时有以下特点。

- 8 位或 9 位数据负载。
- 奇、偶，或无奇偶校验。
- 可配置起始或停止位位数。
- 可配置最高位还是最低位先发送。
- 独立地接收和发送中断。
- 独立地接收和发送 DMA 触发。
- 奇偶错误与帧格式错误状态指示。

(2) USART 寄存器。

- UxCSR：USART x UART 及 SPI 控制与状态寄存器。
- UxUCR：USART x UART 控制寄存器。
- UxGCR：USART x 处理方式控制寄存器。
- UxDBUF：USART x 接收/发送数据缓存区。
- UxBAUD：USART x 波特率控制寄存器。

▶ 3.5.4　CC2530 无线收发器

CC2530 接收器是一款中低频接收器。接收到的射频信号首先被一个低噪放大器(LNA)放大，并把同相正交信号下变频到中频(2MHz)，接着复合的同相正交信号被滤波放大，再通过 AD 转换器转换成数字信号，其中，自动增益控制、最后的信道滤波、扩频、相关标识位、同步字节都是以数字方法实现的。

CC2530 收发器通过直接上变频器来完成发送，待发送的数据存在一个 128B 的 FIFO 发送单元(与 FIFO 接收单元相互独立)中，其中，帧头和帧标识符由硬件自动添加上去。按照 IEEE 802.15.4 中的扩展顺序，每一个字符(4b)都被扩展成 32 个码片，并被送到数模转换器以模拟信号的方式输出。

一个模拟低通滤波器把信号传递到积分(quadrature)上变频混频器，得到的射频信号被功率放大器(PA)放大，并被送到天线匹配。

▶ 3.5.5　CC2530 开发环境

典型的 CVT-WSN-Ⅱ/S 教学实验系统使用的软件比较多，主要包括 CC25XX 无线单片机软件集成开发环境、CC25XX 芯片 Flash 编程软件、ZigBee 协议分析软件、基于 PC 的管理分析软件等。

1. IAR Embedded Workbench for 8051

IAR 嵌入式集成开发环境是 IAR 系统公司设计用于处理器软件开发的集成软件包，具有软件编辑、编译、连接、调试等功能。它包括用于 ARM 软件开发的集成开发环境(IAR Embedded Workbench for ARM)、用于 Atmel 公司单片机软件开发的集成开发环境(IAR Embedded Workbench for AVR)、用于兼容 8051 处理器软件开发的集成开发环境(IAR Embedded Workbench for 8051)、用于 TI 公司的 CC24XX 及 CC25XX 家族无线单片机的底层软件开发、ZigBee 协议的移植、应用程序的开发等。

软件 IAR 8.1.0 可执行文件 EW8051-EV-Web-8101.exe ，按照软件说明书进行正确的安装，保证正常运行。

2. SmartRF Flash Programmer

SmartRF Flash Programmer 用于无线单片机 CC2530 的程序烧写，或用于 USB 接口的 MCU 固件编程，读写 IEEE 地址等。

软件 SmartRF Flash Programmer 可执行文件 SmartRFProg.exe，按照软件说明书进行正确的安装，保证正常运行。

3. ZigBee 协议监视分析软件

Packet Sniffer 用于 802.15.4/ZigBee 协议监视和分析功能，可以对本地的 ZigBee 网络进行协议监视和分析。

软件 ZStack-CC2530-2.3.1 可执行文件 ZStack-CC2530-2.3.1.exe，按照软件进行协议栈安装，保证能正确运行。

4. LED 自动闪烁实验分析

使用 CC2530 软件开发环境 IAR Embedded Wordbench for MCS-51 新建一个工程，完成自己的设计和调试。有关 IAR 的详细说明文档可浏览 IAR 网站或参考安装文件夹里的支持文档 Chipcon IAR IDE usermanual_l_22.pdf。这里仅通过一个简单的 LED 闪灯测试程序带领用户逐步熟悉 IAR for 51 工作环境。在这个测试程序中所需要的工具和硬件是 DTD243A_Demo 仿真器和一个 CC2430 模块 DTD243A。

1) CC2530 的 GPIO 接口

CC2530 单片机具有 21 个数字输入/输出引脚，可以配置为通用数字 I/O 或外设 I/O 信号，可配置为连接到 ADC、定时器、SPI 或串口外设。这些 I/O 口的用途，可以通过用户软件配置一系列寄存器实现。

2) 寄存器

本实验用到 P0 口和 P2 口，两个口的设置类似。以下以 P0 口为例，寄存器主要有 P0(数据)、P0SEL(功能选择)、P0DIR(方向选择)和 P0INP(输入模式选择)；每个寄存器都可以位寻址，表 3-9～表 3-12 列出了各个寄存器的定义和复位值。

表 3-9　P0(P0 口寄存器)

位号	位名	复位值	操作性	描　述
7:0	P0[7:0]	0xFF	读/写	P0 端口普通功能寄存器，可位寻址

复位后 P0=0xFF，对 P0 口进行操作前，一般要先设置好 P0SEL、P0DIR 和 P0INP 寄存器。

表 3-10　P0SEL (P0 功能选择寄存器)

位号	位名	复位值	操作性	描　述
7:0	SELP0_[7:0]	0x00	读/写	0：普通 I/O；1：外设功能

复位后 P0SEL=0x00，即 P0 口为普通 I/O 口。如果要为外设功能，把相应位设为 1 即可。外设功能主要包括 ADC 转换、串口、SPI、定时器、DEBUG 调试口等。

表 3-11　P0DIR (P0 方向选择寄存器)

位号	位名	复位值	操作性	描　述
7:0	DIRP0_[7:0]	0x00	读/写	0：输入；1：输出

复位后 P0DIR=0x00，即 P0 口为输入。如果要为输出，把相应位设为 1 即可。

表 3-12　P0INP（P0 输入模式选择寄存器）

位号	位名	复位值	操作性	描　述
7:0	MDP0_[7:0]	0x00	读/写	0：上拉/下拉，由 P2INP 指定；1：三态

复位后 P0INP＝0x00，即 P0 口为上拉/下拉，具体由 P2INP 寄存器的位 PDUP0 指定：PDUP0＝0 为上拉；PDUP0＝1 为下拉。如果要为三态（高电平、低电平、高阻抗），把相应位设为 1 即可。

3）相关电路图

板上有一个电源灯（D4）、两个状态灯（D2 和 D3），电路如图 3-23 所示。

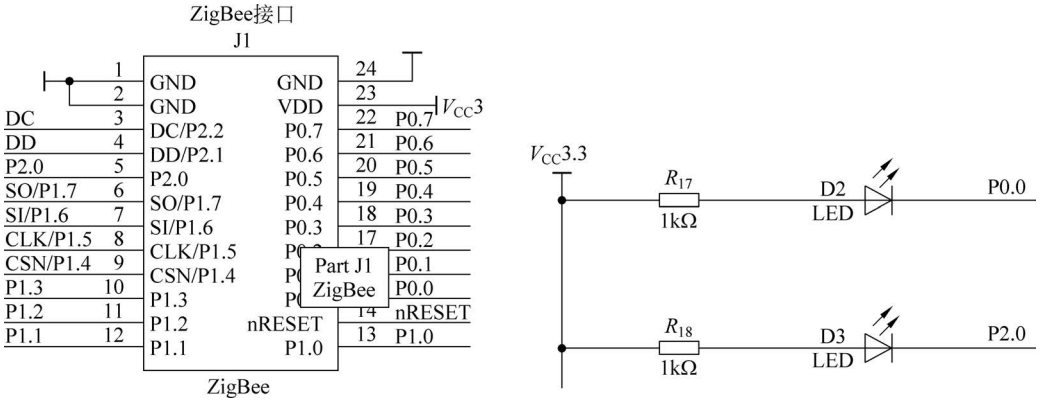


图 3-23　发光二极管驱动电路

当 P0.0 输出低电平时，发光二极管 D2 点亮；P0.0 输出高电平时，发光二极管 D2 熄灭。

当 P2.0 输出低电平时，发光二极管 D3 点亮；P2.0 输出高电平时，发光二极管 D3 熄灭。

4）程序流程及核心代码

（1）程序流程如图 3-24 所示。

（2）核心代码。

```
void main(void)
{
  P0DIR |= 0x01;   //设置 P1.0 为输出方式，复位后为普通 I/O 输入口
  P2DIR |= 0x01;   //设置 P2.0 为输出方式，复位后为普通 I/O 输入口

  while(1)
  {
     P0_0 = 0;     //点亮 D2 发光二极管
     P2_0 = 0;     //点亮 D3 发光二极管

     delay();      //延时子程序，延时大概 400ms

     P0_0 = 1;     //熄灭 D2 发光二极管
     P2_0 = 1;     //熄灭 D3 发光二极管

     delay();
  }
} //end of main()
```

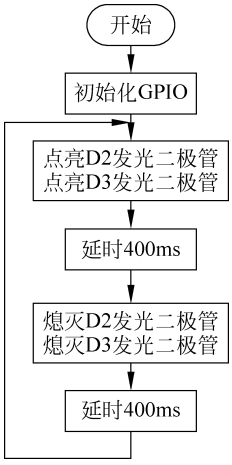


图 3-24　程序流程

5）实验现象

发光二极管 D2 和 D3 同时点亮，一段时间后又同时熄灭，如此循环往复。

3.6 WSN 仿真

▶ 3.6.1 WSN 仿真的特点

无线传感器网络是高度面向应用的网络类型，并且无线传感器网络相对于其他类型的网络有许多限制和独特之处，因此其仿真特点与现有的有线和无线网络有所不同，具有如下特点。

1. 仿真规模大

对于传统的有线网络，利用有限的具有代表性的节点拓扑就可以相当大程度地模拟整个网络的性能。但是对于无线传感器网络，部署于监测区域内的无线传感器节点数目庞大，节点分布密集，网络规模大且具有复杂的、动态变化的拓扑结构，因此无法用有限的节点数目来分析其整体性能，在仿真时必须考虑大规模的网络仿真，并保持一定的仿真效率。

2. 仿真目标不同

传统的有线和无线网络仿真主要分析网络的吞吐量、端对端延迟和丢包率等 QoS 指标，而这些在大部分无线传感器网络的应用中都不是最主要的分析目标。无线传感器网络是以数据为中心的全分布式网络，单个传感器节点的信息意义不大，要求仿真中对整个网络进行协同分析。传感器节点一般采用电池供电并部署于不可更换电池的环境中，要求节点使用寿命长，因此节点的寿命、能耗分析等成为仿真中非常重要的仿真分析目标。

3. 业务模型不固定

无线传感器网络是高度面向应用的，不同的应用有不同的业务模型，也会有不同的事件类型，另外在不同的应用情况下网络的生命周期也不同，因此需要建立一种适合的无线传感器网络业务模型。

4. 节点的特点

传感器节点具有感知物理世界的能力，它对外部突发事件具有很高的灵敏度，而且传感器节点可能会移动或者受到噪声、干扰、人为破坏等因素的影响而失效，也会在不同的工作状态下不断变化，因此在仿真过程中需要考虑传感器节点的特殊性，建立对物理环境、状态变化等的动态模型。

5. WSN 的其他特点

除了以上几个方面以外，无线传感器网络还有许多独特性，例如，硬件平台的多样性、网络节点自身操作系统的特殊性、没有标准的通信协议等，因此在仿真中需要根据应用需求建立适合的仿真模型。

▶ 3.6.2 通用网络仿真平台

随着无线传感器网络的快速发展，对无线传感器网络仿真的研究也有了很多科研成果。目前，不仅有对以前成熟的网络仿真平台的改进，使得它们支持无线传感器网络的新特性，例如 NS-2、OPNET 等，也有在以前平台的基础上开发的仿真工具，例如 SensorSim、SENSE、LSU SensorSimulator 等，还有重新开发的一些面向无线传感器网络的仿真工具，例如 TOSSIM 等。基于无线传感器网络的特点，目前的网络模型并不能完全仿真所遇到的问题。不同的仿真器存在不同的问题，例如过度简化的模型，缺少用户自定义，很难获取现有协议，成本高等。

由于仿真本身就不能达到百分之百的完美，而且目前有很多适用于仿真不同应用场景的传感器网络仿真工具，所以对于一个开发者来说，选择一款适合自己项目的仿真器是很重要的。但是如果没有对现有仿真器全面认识，要做出这样的选择是很困难的。此外，仿真器开发者通过对现有仿真工具的了解和比较，还可以认识到现有仿真工具及自己开发的模型的优缺点，这对仿真器的优化是有很大帮助的。因此，对现有的、优秀的仿真工具做一下详细了解是很有必要的。在此介绍几种常用的仿真工具，描述它们各自的特性及优缺点，为无线传感器网络仿真平台的选择及设计提供指导。

1. NS-2

NS-2(Network Simulator Version 2)是无线传感器网络中最流行的仿真工具。它起源于1989年为通用网络仿真设计的NS(Network Simulator)，是一个开源的面向对象离散事件仿真器，采用模块化方法实现。用户可以通过“继承”来开发自己的模块，具有很好的扩展性，能够对仿真模型进行扩展，也可以直接创建和使用新的协议。NS-2通过C++与OTcl的结合来实现仿真，C++用于实现协议及对NS-2模型库的扩展，OTcl用于创建和控制仿真环境，包括选择输出数据。NS-2包括大量的协议、通信产生器(Traffic Generator)及工具。

NS-2对无线传感器网络的仿真是对Ad Hoc仿真工具的改进并添加一些组件来实现的。对传感器网络仿真的支持包括传感信道、传感器模型、电池模型、针对无线传感器的轻量级协议栈、混合仿真及场景生成等。通过对NS-2的扩展，使得它能够通过外部环境来触发事件。NS-2的扩展性和面向对象设计使得新协议的开发和使用非常简便，而且有许多协议可以免费获取，还可以将自己的仿真结果跟其他人的算法比较，这更加促进了NS-2的发展。

NS-2也存在以下缺点。

(1) 不适合大规模的无线传感器网络仿真。由于是面向对象的设计，使得在仿真大量节点的环境时性能很差。在仿真中，每个节点都是一个对象，并能够与其他任何节点交互，在仿真时会产生大量的相关性检查。

(2) 缺少用户自定义。包格式、能量模型、MAC协议，以及感知硬件模型都与大部分传感器中的不同。

(3) 缺少应用模型。在许多网络环境中这不是问题，但是传感器网络一般包括应用层与网络协议层之间的交互。

(4) 使用比较困难。NS-2入门难，并且有很多版本，使得仿真结果之间的比较变得困难。

2. OPNET

OPNET(Optimized Network Engineering Tool)是一个面向对象的离散事件通用网络仿真器，它使用分层模型来定义系统的每一方面。顶层包括网络模型，在此定义拓扑结构；第二层是节点，在此定义数据流模型；第三层是过程编辑器，它处理控制流模型；最后包含一个参数编辑器支持上面的三层。分层模型的结果是一个为了离散事件仿真引擎的事件队列和一系列代表在仿真中处理事件的节点的实体集合，每个实体包括一个仿真时处理事件的有限状态机(Finite State Machine，FSM)。OPNET能够记录大量用户定义结果的集合，支持不同特定传感器硬件的建模，例如物理连接收发器和天线，能自定义包格式。仿真器通过一个图形用户接口帮助用户开发不同的模型，整个接口可以模拟、图形化、动画展示输出结果。OPNET和NS-2一样有相同的面向对象的规模问题。它并不像NS-2及GloMoSim那样流行，它为商业软件，在获得协议方面也不如NS-2方便。

OPNET的特点如下。

(1) 建模与仿真周期：OPNET 提供了强大的工具，以帮助用户完成设计周期中 5 个设计阶段的 3 个，即创建模型、执行仿真和结果分析，如图 3-25 所示。

(2) 分层建模：OPNET 使用层次结构建模，每个层次描述仿真模型的不同方面，共有 4 种编辑器，即网络编辑器、节点编辑器、进程编辑器和参数编辑器。

(3) 专用于通信网络：详细的模型库提供对现有协议的支持，并且允许用户修改现有模型或者创建自己的模型。

(4) 自动仿真生成：OPNET 模型能被编译为可执行代码，然后调试和执行，并且输出数据。它拥有探测器编辑器、分析工具、过滤工具、动画视图等结果分析工具。

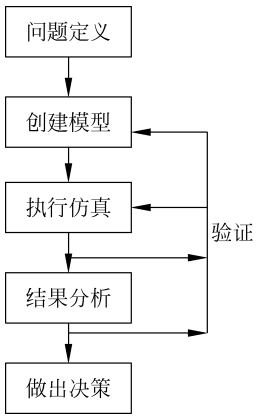


图 3-25　建模与仿真周期

3. OMNet++

OMNet++(Objective Modular Network Testbed in C++)是一个开源的、面向对象的离散事件仿真器，一般用来仿真通信网络及其他一些分布式系统。OMNet++由多层嵌套的模块构成，如图 3-26 所示，模块分为简单模块和复合模块。简单模块用于定义算法及组成底层；复杂模块由多个简单模块组成，这些简单模块之间使用消息进行交互。顶层模块称为系统模块(System Module)或者网络，它包括一个或者多个子模块，每个模块又可以嵌套子模块，对嵌套的深度没有限制。

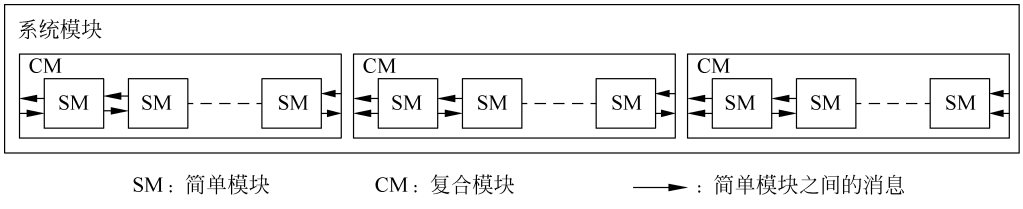


图 3-26　OMNet++模块

OMNet++的传感器网络模块称为 SensorSim，传感器节点是一个复合模块，它包括三类模块，分别为代表每个协议层的模块、反映硬件的模块和一个协调器模块。协调器模块负责协议层和硬件之间的通信，为两者传递必需的消息。在传感器节点模块之外有代表被感知物理对象的模块。传感信道与传感器节点通信，无线信道与网络通信。SensorSim 是基于组件实现的，它比 NS-2 具有更高的效率，能够准确地模拟大部分硬件，包括对物理环境的建模，协议栈的所有层都可以修改。

OMNet++有以下缺点。

(1) 设计方法与其他仿真器大不相同，学习困难。

(2) 使用 SensorSim 所做的研究发布很少。

(3) 实现的协议很少。

4. GloMoSim

GloMoSim(Global Mobile Information Systems Simulation Library)于 1998 年针对移动无线网络开发，具有以下特点。

(1) 并行仿真：GloMoSim 采用 Parsec 语言(C 语言的扩展，支持并行编程)实现，能够实现并行仿真。

(2) 可扩展性：GloMoSim 库中所有的协议均以模块的形式存在。GloMoSim 的结构包

括多层，如图 3-27 所示，每层都使用不同的协议集合并有一个与相邻层通信的应用程序接口（Application Program Interface，API）。

（3）面向对象：GloMoSim 采用面向对象方法实现，但是将节点划分为多个对象，每个对象负责协议栈中的一层，减轻了大型网络的开销。

GloMoSim 仍然存在以下问题。

（1）仿真网络类型限制。GloMoSim 在仿真 IP 网络时很有效，但是不能仿真其他类型的网络，因此有很多传感器网络都不能在 GloMoSim 上仿真。

（2）不支持外部环境事件。GloMoSim 不支持仿真环境之外的环境事件，所有的事件都必须由网络内部的其他节点产生。

（3）不再更新。在 2000 年之后，GloMoSim 已经停止更新，并被其商业版本 QualNet 取代。

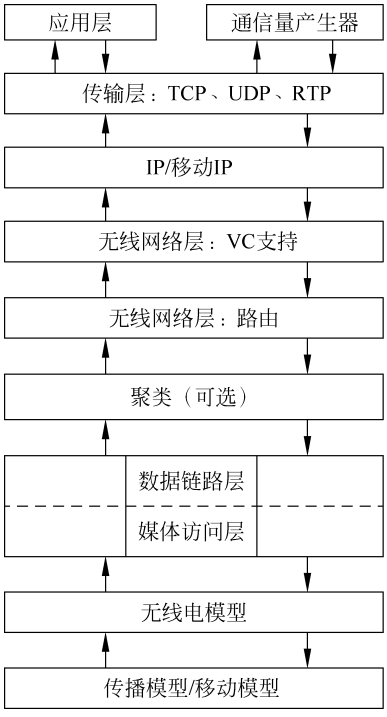


图 3-27　GloMoSim 的结构

5. QualNet

QualNet 是 GloMoSim 的商业版本，它对 GloMoSim 做了许多扩展，使其包括许多针对有线及无线网络（包括局域网、Ad Hoc 网络、卫星网络和蜂窝网等）的模型、协议集合、文档及技术支持。QualNet 包括三个库，即标准库、MANNET 库和 QoS 库。标准库提供有线及无线网络中大部分的模型及协议；MANNET 库提供标准库中不包括的、针对 Ad Hoc 网络的特定组件；QoS 库提供针对 QoS 的协议。这三种库均是以二进制代码格式提供的，用户可以使用配置文件修改其行为，但是不能更改其核心功能。QualNet 包括 5 个图形用户界面模块，即场景设计器（Scenario Designer）、图形生成器（Animator）、协议设计器（Protocol Designer）、分析器（Analyzer）及包跟踪器（Packet Tracer）。场景设计器及图形生成器是图形化的试验设置和管理工具，可以直观地单击及拖曳工具，定义网络节点的地理分布、物理连接和功能参数，定义每个节点的网络层协议和话务特征；协议设计器是用于定制协议建模的有限状态机（Finite State Machine，FSM）工具，利用直观的、基于状态的可视工具定义协议模型的时间和过程，缩短开发时间，可以修改现有协议模型，或者自己定制协议和特定的统计报告；分析器是统计图形表示工具，可以在上百个预设计报告中选择或定制自己的报告；包跟踪器是用于查看数据分组在协议栈中发送和接收过程的分组级可视化工具。这些模块合理、有效结合，使得 QualNet 是一个非常完善的模拟器。

6. Matlab

Matlab（Matrix Laboratory）是一种科学计算软件，以矩阵形式处理数据。Matlab 将高性能的数值计算和可视化集成在一起，并提供大量的内置函数，可用于通信系统设计与仿真。Matlab 提供基本的数学算法、集成 2D 和 3D 图形功能，并提供一种交互式的高级编程语言。Matlab 还有一系列的专业工具箱和框图设计环境 Simulink。Simulink 用来对各种动态系统进行建模、分析和仿真，可以对任何一个能够用数学描述的系统进行建模，包括连续、离散、条件执行、事件驱动、单速率、多速率和混杂系统等。Simulink 利用鼠标拖放的方法建立系统框图模型的图形界面，提供了丰富的功能块以及不同的专业模块集合，利用 Simulink 几乎可以

做到不书写一行代码完成整个动态系统的建模工作，简化了系统设计及仿真。

WiSNAP(Wireless Image Sensor Network Application Platform)是一个针对无线图像传感器网络(Wireless Image Sensor Network)设计的基于 Matlab 的应用开发平台。它使得研究者能够使用实际的目标硬件来研究、设计、评估算法及应用程序。WiSNAP 还提供了标准、易用的应用程序接口(Application Program Interface，API)来控制图像传感器及无线传感器节点，不需要详细了解硬件平台。开放的系统结构还支持虚拟的传感器和无线传感器节点。

WiSNAP 的程序结构如图 3-28 所示，它为用户提供了两类应用程序接口，即无线传感器节点应用程序接口及图像传感器应用程序接口。用户可以很方便地控制无线传感器节点和图像传感器，实现应用程序开发。WiSNAP 的设备库(包括 CC2420DB 库、ADCM-1670 库和 ADNS3060 库)能够实现特定硬件的协议及功能，可以用 Matlab 脚本和 Matlab 可执行文件实现。操作系统则提供对计算机硬件(包括串口、并口)的访问。

图 3-28　WiSNAP 的程序结构

7. J-Sim

J-Sim 是采用 Java 语言实现的通用仿真器，它使用基于组件结构的设计方法、增强的能量模型，能够仿真传感器对环境的检测。J-Sim 的组件结构如图 3-29 所示，目标节点产生激励，传感器节点响应激励，汇聚节点是报告激励的目标，每个组件又被分解为不同的部分，使得容易使用不同的协议。J-Sim 可以仿真应用程序，可以连接到实际的硬件，但是使用较难。

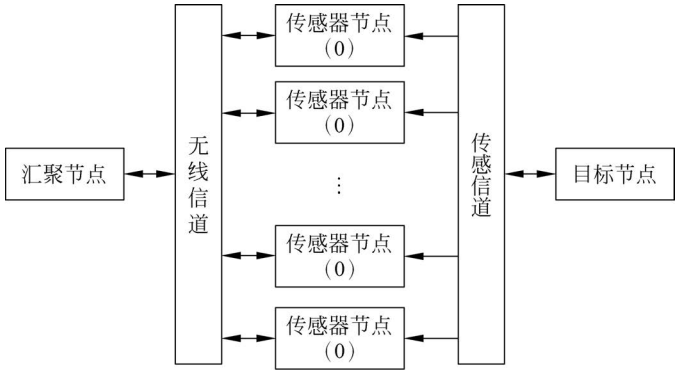


图 3-29　J-Sim 的组件结构

▶ 3.6.3　针对 WSN 的仿真平台

1. TOSSIM、TOSSF 与 SENS

TOSSIM、TOSSF 与 SENS 均能够对 TinyOS 程序进行仿真。TOSSIM(TinyOS Simulator)是为运行于 MICA 系列传感器节点的 TinyOS 应用程序所设计的,它与 TinyOS 一起发行,包括一个可与仿真交互的可视化仿真过程图形用户界面 TinyViz(TinyOS Visualizer)。TOSSIM 在设计时主要考虑以下 4 个方面。

(1) 规模:系统应该能够处理拥有不同网络配置的若干节点。

(2) 完整性:为了准确地捕获行为,TOSSIM 包括尽量多的系统交互。

(3) 保真度:如果要测试准确,需要捕获很细小的交互。

(4) 桥接:桥接 TOSSIM 之间的差距,测试及验证在实际硬件中执行的代码。

针对大规模仿真的目标,仿真器中的每个节点与一个有向图(每条边都有一个概率比特误码)连接。为了保证传输的有效性,用"0"表示比特错误,但会根据情况的不同而改变。此外,所有的节点运行相同的代码,这样可以提高效率。

TOSSIM 由不同的组件组成,其结构如图 3-30 所示,支持编译网络拓扑图、离散事件队列、被模拟的硬件、通信基础结构(允许仿真器与外部程序通信)。大部分应用程序代码都不用改变,只是在与硬件交互的应用程序场合有所区别。

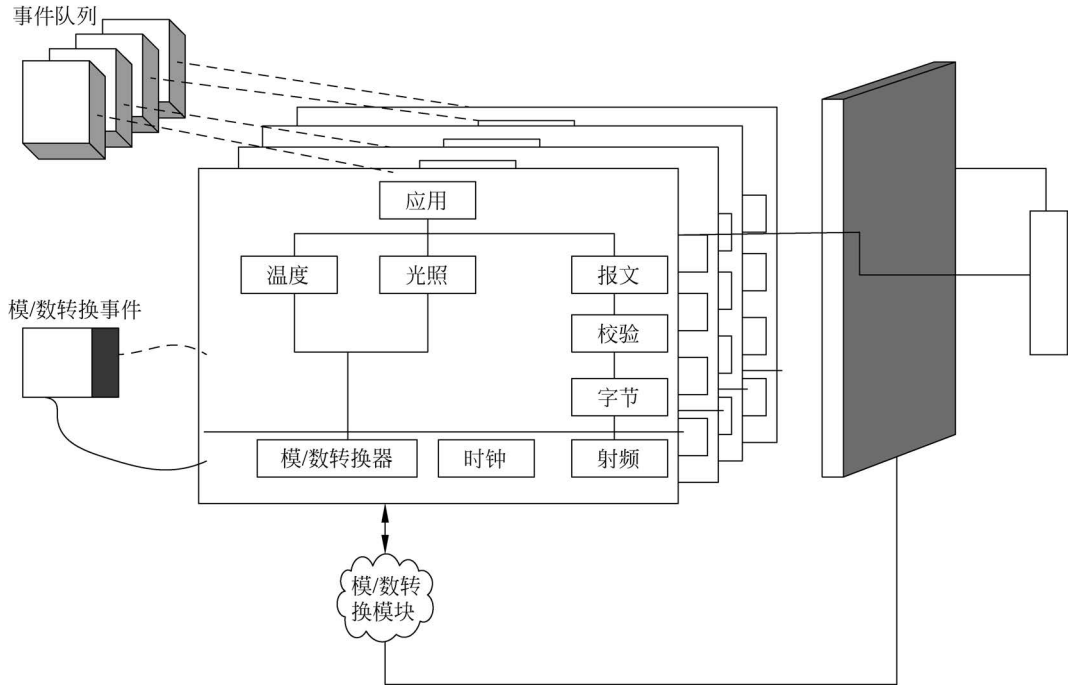


图 3-30　TOSSIM 的结构

TOSSIM 的概率比特误码模型会导致错误,并在分析低级协议时降低了仿真器的效率。在编译时,微小的时序和中断属性丢失会影响与其他节点的交互,也同样会降低准确度。另外,TOSSIM 只考虑了传感器的数据采集硬件的仿真,并没有实现对环境触发的反应的仿真。

PowerTOSSIM 是一个电源模型,已经集成到 TOSSIM 中。PowerTOSSIM 对 TinyOS 应用程序消耗的电能进行建模,包括 Mica2 传感器节点能量消耗的详细模型。

TOSSF 是一个可升级的仿真框架，它是在 DaSSF（the Dartmouth Scalable Simulation Framework）和 SWAN（Simulator for Wireless Ad hoc Networks）的基础上开发的，其结构如图 3-31 所示。DaSSF 是一个拥有高性能和仿真规模的、改进的、优化的仿真内核。SWAN 是一系列在 DaSSF 内核上构建的 C++类的集合，它为无线自组织网络的仿真提供许多模型。SWAN 提供运行时的模块配置扩展性。

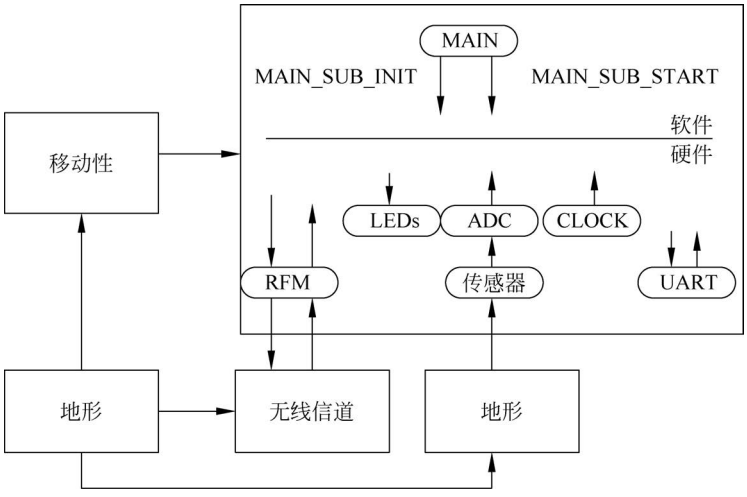


图 3-31　TOSSF 的结构

对于熟悉 TinyOS 编程的人来说，TOSSF 的使用非常容易，只需要学习编写基本的 DML（Domain Modeling Language）配置脚本，以定义仿真场景和仿真节点。TOSSF 也有一定的限制：

- 所有的中断都是在任务、命令或者事件执行完之后才得到响应的。
- 命令和事件处理程序在零仿真时间单元执行。
- 没有抢占。

SENS 是一个可定制的传感器网络仿真器，它包括针对应用程序、网络通信、物理环境的可互换、可扩展的组件。

SENS 拥有一个可定制组件的分层结构，具有平台无关性，添加新的平台只需要添加相应的配置参数即可。SENS 采用新颖的物理环境建模机制，将环境定义为一些可交换的单元的格子。

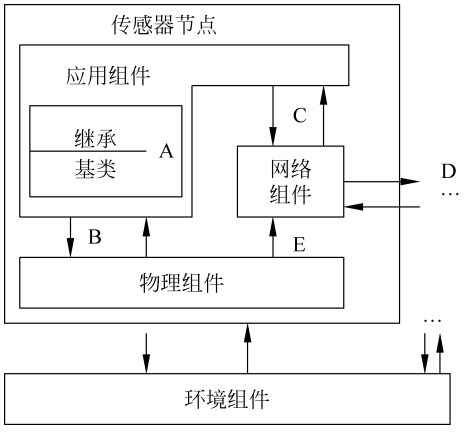


图 3-32　SENS 的结构

SENS 的结构如图 3-32 所示，包括多个模拟的传感器节点和一个环境组件，每个节点包括三部分，即物理组件、网络组件及应用组件，每个组件有一个虚拟时钟，消息能以任何延时发送。用户可以使用 SENS 提供的组件，也可以对现有组件进行修改，甚至自己编写新的组件。通过选择不同的组件组合，用户可以实现不同的网络应用。节点也可以有不同的配置，从而可以仿真不同种类的传感器网络，这在不同节点具有不同的能力及添加新节点的情况下是很有用的。

（1）应用组件：模拟单个传感器节点上软件的执行，它与网络组件通信以接收和发送数据包，与

物理组件通信以读取传感器数据和控制执行机构。SENS 可以通过继承 Application 类来完成应用开发，但是不能直接运行在现有硬件平台上。另外，SENS 提供了一个瘦兼容层，使得仿真器与实际传感器节点可以使用相同的代码，从而能够仿真 TinyOS 应用程序。

(2) 网络组件：模拟传感器节点数据包的接收和发送功能。所有网络组件继承自 Network 基类(指定基本的网络接口)。每个网络组件与一个应用组件和相邻节点的网络组件相连。相邻节点之间交换的消息格式是固定的，从而实现不同特性的网络。网络模型有 SimpleNetwork、ProbLossyNetwork 和 CollisionLossyNetwork 三种。SimpleNetwork 简单地将消息发送给邻居节点并将接收到的消息传递给应用组件；ProbLossyNetwork 按照一定的错误概率传递或者丢弃数据包；CollisionLossyNetwork 在接收节点处计算数据包的碰撞。

(3) 物理组件：模拟传感器、执行机构、电源、节点的电能消耗及与环境的交互。

(4) 环境组件：模拟传感器及执行机构可能所处的实际环境。

Tython 是实现基于 Python 脚本的、对 TOSSIM 仿真器的扩展，Tython 与 TOSSIM 的设计对应如图 3-33 所示。Tython 包括一个丰富的脚本原语库，能够让用户描述动态的、能重复使用的仿真场景。利用 TinyOS 事件驱动的优点，允许用户附加脚本反馈到特定的仿真场景。脚本也可以在整个网络和节点级分析并改变环境变化反应的行为。

2. ATEMU

ATEMU 弥补了 TOSSIM 的不足，和 TOSSIM 一样，ATEMU 的代码是与 Mica2 平台兼容的二进制代码。ATEMU 使用逐个周期策略运行应用程序代码，仿真比 TOSSIM 更加细致。这是通过对 Mica 使用的 AVR 处理器的仿真来实现的。

ATEMU 使用 XML 配置文件对网络进行配置。这使得网络以分等级的方式被定义，顶层定义网络特性，下面的层定义每个节点的特性。ATEMU 的结构如图 3-34 所示，它提供一个称为 XATDB 的图形用户接口，被用来调试和观察代码的执行，允许设置断点、单步调试及其他的调试功能。

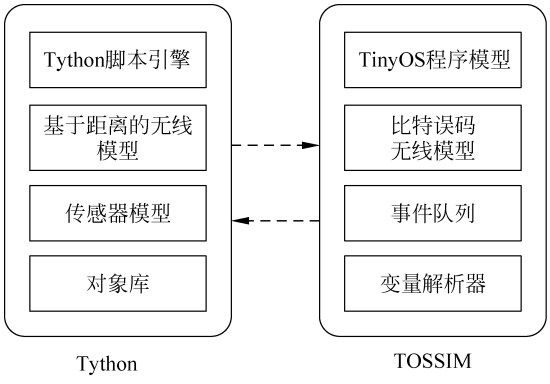


图 3-33　Tython 与 TOSSIM 的设计对应

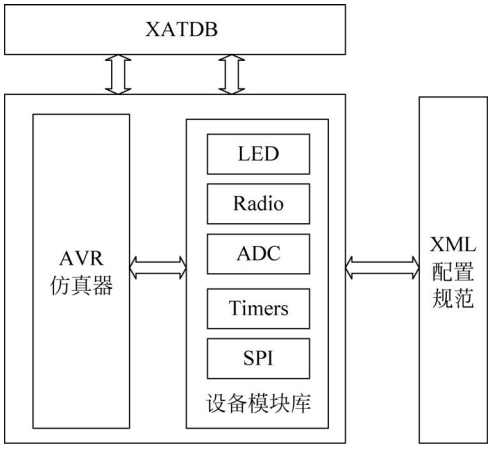


图 3-34　ATEMU 的结构

ATEMU 提供了一个精确的仿真模型，在此每个 Mica 传感器节点能够运行不同的应用程序代码。它比 TOSSIM 准确，但是速度和仿真规模有所降低，最多能准确仿真 120 个节点。除了逐条指令译码带来的开销以外，ATEMU 还有面向对象模型所带来的开销，一个无线传输会影响网络中的其他节点。除了仿真规模问题之外，ATEMU 是最准确的传感器仿真器之一。

3. Avrora

Avrora是一个周期准确的指令级传感器网络仿真器，它能对10 000个节点进行仿真，并比具有同样准确度的仿真器约快20倍，能实时处理25个节点。

Avrora是一个试图在TOSSIM及ATEMU两者之间找到平衡点的新仿真器(Emulator)，它采用Java实现，而TOSSIM和ATEMU都采用C语言实现。和许多面向对象仿真器一样，Avrora将每个节点作为一个线程，但是它仍然运行实际的Mica代码。和ATEMU一样，Avrora以逐条指令方式执行代码，但是为了获得更好的规模和速度，所有的节点在每条指令后都没有进行同步处理。

Avrora使用两种不同的同步策略。第一种方法使用一个同步间隔来定义同步发生的频率，这个值越大，同步间隔越大，同步间隔的“1”和ATEMU大致相同。注意，不能把这个值设得太高，否则节点将会运行超过其他节点事件影响的时间。第二种方法是在同步前等待邻居节点达到一个特定的仿真时间，一个全局数据结构用来保存每个仿真器的本地时间，该算法允许每个节点比其他的节点仿真时间超前直到需要同步。通过减少同步，Avrora有效地减少了开销。

4. SENSE

SENSE(Sensor Network Simulator and Emulator)是在COST (Component Oriented Simulation Toolkit，一个通用离散事件仿真器) 之上开发的，编程语言为C++。SENSE的设计受三个仿真工具的影响，它具有类似NS-2的功能，和J-Sim一样采用基于组件的结构，和GloMoSim一样支持并行仿真。由于采用基于组件的结构和并行仿真，开发者可以将重点放在仿真中的三个重要因素上，即可扩展性、重用性、仿真规模。由于采用了基于组件的仿真并考虑到了存储器的有效使用，以及传感器网络特定模型等实际问题，使得SENSE成为无线传感器网络研究中简单、有效的仿真工具。

1) 扩展性

基于组件的仿真使得SENSE具有足够的扩展性。

组件端口模型：如图3-35所示，使得仿真模型容易扩展，如果有一致的接口，则新的组件可以代替旧的组件，不需要继承。

仿真组件分类：使得仿真引擎具有扩展性，高级用户可以开发满足特定需求的仿真引擎。

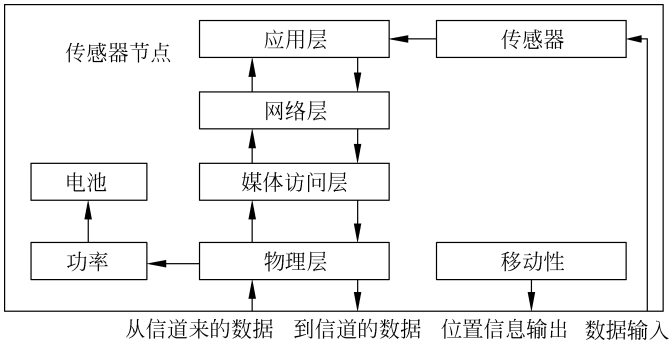


图3-35 传感器节点的内部结构

2) 重用性

模块之间的依赖性降低，从而增加了它们的可重用性。在一个仿真中实现的组件，如果满足另外一个仿真的接口及语义需求，则可以被重用。

C++模板的扩展同样提高了可重用性。组件通常定义为模板类，以处理不同类型的数据。

3）并行仿真

SENSE 提供了并行仿真和串行仿真可选方式，其系统默认使用串行仿真。

4）用户

在 SENSE 中包括三类用户，即高级用户、网络构建者和组件设计者。

高级用户不需要什么编程技巧，只需要选择合适的模型和模板并设置相应的参数，便可以构建传感器网络的仿真。它们不关心扩展性及重用性，但是要求仿真可以升级。

网络构建者需要构建新的网络拓扑等，它们依赖于现有模块以构建网络模型，主要关心可重用性。

组件设计者需要修改模块或者构建新的模块，主要关心可扩展性。

5. Sidh

Sidh 是一个采用 Java 语言实现的、基于组件的、专门为无线网络设计的仿真器。它由许多模块组成，模块间通过事件交互。每个模块通过一个接口来定义，该接口可以与其他模块交互，只要模块符合特定接口就可以被用在仿真器中。

模块包括不同的种类，有仿真器、事件、媒介、传播模型、环境、节点、处理器、无线收发器、传感器与执行机构、电源、物理层协议、MAC 层协议、路由协议和应用层。仿真器模块是一个离散事件仿真器，它是 Sidh 的基础；事件模块负责模块间通信，包括一个指定事件发生时间的仿真时间；媒介模块指定无线媒介的属性，在每个节点上保持位置和无线电属性；传播模型定义发射机与接收机之间的信号强度；环境模块跟媒介模块类似，但它是模拟物理环境；节点模块代表传感器节点，包括组成传感器节点的所有模块，如硬件模块、协议模块和应用层模块；处理器模拟处理器的工作状态及每种状态下的能量消耗；无线收发器模拟无线收发器的状态、相关行为及能量消耗；传感器与执行机构跟无线收发器类似，最大的区别是它们是与环境接口而不是媒介；电源模块模拟每个节点的电源供应；物理层是网络栈的底层，其提供的服务有无线收发器状态的改变、载波侦听或者空闲信道评估、发送和接收数据包、接收能量检测、多信道方式下的物理信道选择；MAC 层协议在物理层之上，其提供的服务有 MAC 层状态的改变、设置或获取协议参数、发送和接收数据包，Sidh 实现了多个 MAC 层协议，如 CSMA、Bel、B-MAC、TRAMA 等；路由协议在 MAC 层之上，其提供不能直接通信的节点之间的多跳路由服务；应用层驻留在网络栈的上面，与底层协议、传感器与执行机构接口以实现完整的无线传感器网络应用。

Sidh 试图创建接近现实传感器的仿真器，它能够很容易地代替或者交换任何层次的模块，但这是以牺牲效率为代价的。

6. EmStar

EmStar 是一个基于 Linux 的框架，它有多种运行环境，从纯粹的仿真到实际部署。在每种环境下均使用相同的代码和配置文件，这使开发变得容易。和 SensorSim 一样，它在仿真时提供一个选项与实际硬件接口。EmStar 包括一系列的工具，其中 EmSim、EmCee 可以实现仿真，它们包括几个精确度体制，支持不同精度级别的透明仿真，加速了开发及调试。EmSim 在模拟了无线收发器及传感信道的简单仿真环境中并行地运行许多节点。EmCee 运行 EmSim 核，但是提供一个与实际低功耗无线收发器的接口。EmStar 源代码和配置文件与实际部署系统的一样，可以减少开发及调试过程中的工作。EmStar 的仿真模型是一个基于组件的离散事件仿真模型，如图 3-36 所示。EmStar 使用了简单的环境模型和网络媒介，所能运行的节点类型有限。

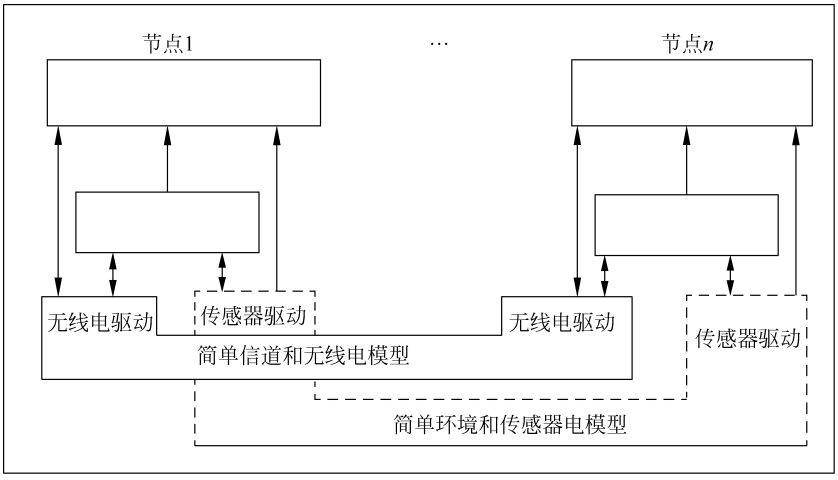


图 3-36　EmStar 仿真模型

7. SimGate

SimGate 是一个 Intel Stargate 设备的全系统仿真器。SimGate 能捕获 Stargate 内部组件的行为，包括处理器、内存、通信(串口和无线)以及外设。SimGate 是一个虚拟设备，它虚拟 Stargate 设备，引导并运行 Linux 操作系统，所有的二进制程序都在 Linux 上运行。

SimGate 能准确地估计处理器周期计数值，而且该功能为了提高仿真性能，能够使得周期更加准确。

SimGate 在一个系统中使用了不同的方法，对设备组件的性能进行评估，包括某些组件的周期级仿真和基准时间选择。SimGate 与实际设备执行同样的操作系统和应用程序二进制代码。

SimGate 能仿真 Stargate 设备的以下特性。

(1) 不带 Thumb 支持、带 XScale DSP 指令的 ARM v5TE 指令集。

(2) XScale 流水线仿真。

(3) PXA255 处理器，包括 MMU、GPIO、中断控制器、实时时钟、操作系统定时器和内存控制器。

(4) 与相连的 Mote 节点通信的串行接口(UART)。

(5) SA-1111 StrongARM 协同芯片。

(6) 64MB SDRAM 芯片。

(7) 32MB Intel StrataFlash 芯片。

(8) 包括 PCMCIA 接口的 Orinoco 无线局域网 PC 卡。

SimGate 还能与 SimMote 结合实现与其他传感器网络结合的仿真。另外，SimGate 还支持调试功能，可以设置断点等。

▶ 3.6.4　WSN 工程测试床

在无线传感器网络中，仿真是一个重要的研究手段。然而，仿真通常仅限于对特定问题的研究，并不能获取节点、网络及无线通信等运行的详细信息，只有实际的测试床才能够捕获到这些信息。虽然在验证大型传感器网络方面存在一些有效的仿真工具，但只有通过对实际的传感器网络测试床的使用才能真正理解资源的限制、通信损失及能源限制等问题。另外，无线

传感器网络的测试是比较困难的，在实验过程中需要对许多节点反复地进行重编程、调试，而且需要获得其中的一些数据信息。通过测试床可以对无线传感器网络的许多问题进行研究，简化系统部署、调试等步骤，使无线传感器网络的研究变得容易。最近几年，随着传感器、无线通信设备等的价格快速降低和尺寸缩小，设计出了很多测试床来研究及验证这些系统的属性。本节主要介绍 MoteLab、GNOMES 及 IBM 的无线传感器网络测试床。

1. MoteLab

MoteLab 是一个基于 Web 的无线传感器网络测试床，它包括一组长期部署的传感器网络节点以及一个中心服务器，其结构如图 3-37 所示。中心服务器处理重编程、数据访问，并提供创建和调度测试床工作的 Web 接口。MoteLab 通过流线型访问大型、固定网络设备，加速了应用程序的部署；通过自动数据访问，允许离线对传感器网络软件的性能进行验证，加速了系统调试及开发；允许本地和远程用户通过 Web 接口接入测试床；通过调度和分配系统能够确保接入用户之间公平地分享资源。MoteLab 已经用于多个项目的研究（如 MoteTrack、CodeBlue），也可用于教学等。

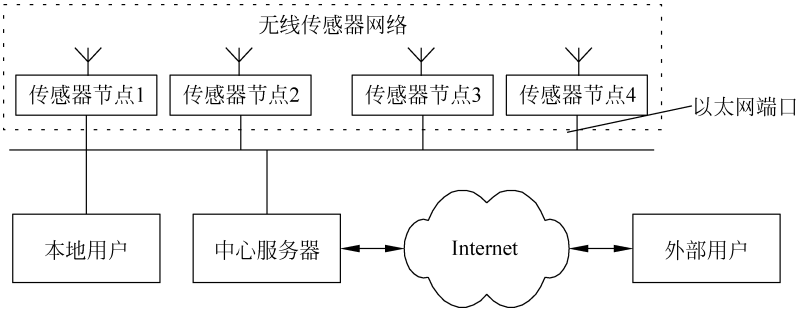


图 3-37　MoteLab 的结构

1）MoteLab 硬件

刚开始，MoteLab 由 26 个 Mica2 节点、26 个以太网接口板（6 个由 Intel 实验室的 Phil Buonadonna 开发的 EPRB 和 20 个克尔斯博公司的 MIB-600）组成，每个 Mica2 节点与一个以太网接口板相连，使得每个节点都连接到局域网上，后来更新为 30 个克尔斯博公司的 MicaZ 节点。

2）MoteLab 软件

MoteLab 软件是一个管理以太网连接的传感器节点的测试床的软件工具集合。中心服务器处理调度、对节点重编程、数据访问、为用户提供 Web 接口。用户通过 Web 浏览器接入测试床，以设置或者调度工作，下载数据。MoteLab 有以下几个软件组件。

（1）MySQL 数据库。MySQL 数据库用于存储测试床运行所需的所有信息，这些信息包括工作产生的数据和测试床的状态数据。每一个用户对应一个数据库，并保存与它相关的工作产生的数据，而每一项工作都会对应一张数据表，另外还有一张表存储与工作有关的消息类型。一个单独的数据库存储测试床的状态信息，包括用户信息和访问权限、节点状态、上传的可执行组件和类文件信息、工作属性、对测试床安排的表述。这些信息提供给其他 MoteLab 组件并可以被它们修改。

（2）Web 接口。MoteLab 采用 PHP 产生动态页面，JavaScript 提供交互式用户体验，用户对测试床的访问具有平台无关性。普通用户可以创建工作、编辑工作、对工作及用户信息进行操作等，系统管理员可以添加、修改用户和对测试床的划分进行配置。

(3) DBLogger。DBLogger 是一个 Java 程序,它在每项工作开始时运行,通过 TCP 端口连接到每个节点,对节点所发的消息进行解析,并将结果存储于数据库中。消息的每个字段都被解析,并对应数据库中的一个字段。另外,数据库表中还添加了三个字段,即产生消息的节点、产生消息的时间和消息的全局编号。

(4) 工作管理后台程序(Job Daemon)。Job Daemon 采用 Perl 脚本实现,负责设置实验,包括节点重编程、启动必需的系统组件及在工作完成后将它们释放。例如,停止节点活动,杀死在工作中使用的进程,将数据从 MySQL 数据库中导出为适合下载的格式。

3) MoteLab 的其他特点

(1) 用户配额。通过一个用户配额,系统使多个用户能够分享测试床,它只控制工作数量,不控制用户数量,即用户在测试床上执行的工作受到限制,而使用测试床并不受到限制。

(2) 直接节点访问。通过 TCP/IP 连接对节点的串口直接进行访问。

(3) 功率测量(In Situ)。在一个节点上连接一个网络化数字万用表(Keithley 2701),并按照 250Hz 连续采样实现功率测量。用户可以下载带有时间的电流数据,以便对网络的功率消耗进行分析。

(4) 两种使用模式。用户有批使用(Batch Use)和实时访问(Real-time Access)两种使用模式,批使用是在无人照顾的情况下执行多个工作;实时访问是用户通过串口或者实时数据库访问直接与正在运行的工作交互。

(5) 适合不同用户。本地用户和外部用户均可使用测试床,外部用户需要通过申请才可以使用。

2. GNOMES

GNOMES 设计用于研究不同种类无线传感器网络的属性,测试无线传感器网络结构中的理论并在实际应用环境中部署。GNOMES 的设计重点为低成本和低功耗,要求能够实现自组织,即协调工作,以使处理特定问题的传感器节点的价格低于 25 美元(批量价格,节点数目需大于 1000),采用了特殊的电源管理及功耗管理,降低系统功耗。

1) 硬件

GNOMES 的节点结构如图 3-38 所示,使用 TI 公司的低功耗处理器 MSP430F14x、32KB I^2C EEPROM,支持多电源(电池及太阳能)、可选的 GPS 模块、传感器模块、蓝牙通信模块及两个可扩展接口。

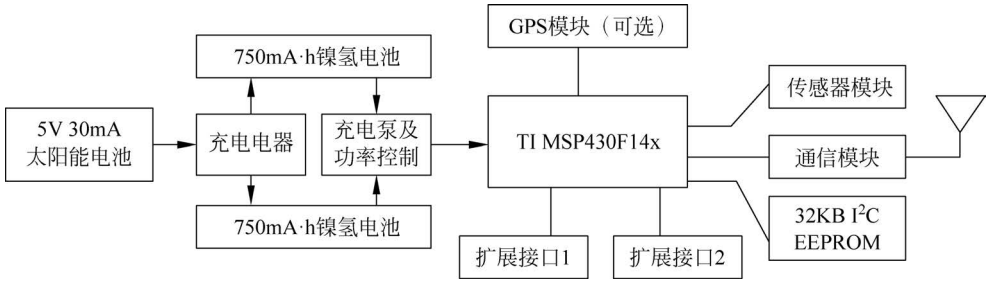


图 3-38　GNOMES 的节点结构

该节点具有以下独特的特点。

(1) 节点使用寿命长。通过双电池(两个 AA 镍氢电池)供电,系统从其中一个电池获得主电源,并通过外部的太阳能电池对未使用的辅助电池进行充电,另外还通过自适应的软件算法监测系统的性能及电池的电压,以决定是否更换电源。

(2) 传感器采用模块化设计,只需要在传感器接口插入不同的传感器模块便可更换节点。

(3) 通信模块也采用模块化设计,目前已有 2.4GHz 蓝牙无线收发模块和 900MHz 无线模块。

2) 软件

软件部分包括 GNOMES 操作系统、驱动程序和公共 API。操作系统实现了进程管理和调度,能够同时执行多个应用程序和后台功率管理。驱动和 API 包括部分蓝牙协议栈的实现、NMEA(National Marine Electronics Association)字符串分析,以更好地管理 GPS 模块的数据、与 EEPROM 通信的 I^2C 驱动、控制 A/D 转换器的接口以及一个功率管理进程。所有的软件均采用 C 语言和汇编语言混合编写,编程环境为 IAR Embedded Workbench。

3) 应用

由于 GNOMES 的低价格和较长的生命周期,它能应用于地震监测、人员跟踪或者机器的行踪跟踪、污染程度考查、腐蚀探测及修复程度监测等。

3. IBM WSN Testbed

IBM 苏黎世研究实验室(Zurich Research Laboratory)开发的测试床是一个完整的端到端解决方案,它包括多种无线传感器网络、一个传感器网关、连接传感器网络及企业网络的中间件和传感器应用软件。这个测试床用于评估与传感器网络相关的无线通信技术(如 IEEE 802.15.4/ZigBee 网络、蓝牙无线局域网和 IEEE 802.11b 无线局域网)的性能,测试在传感器及应用服务器之间实现异步通信的轻量级消息协议以及开发具体的应用(如远程测量和位置感知)。该测试床的体系结构如图 3-39 所示。

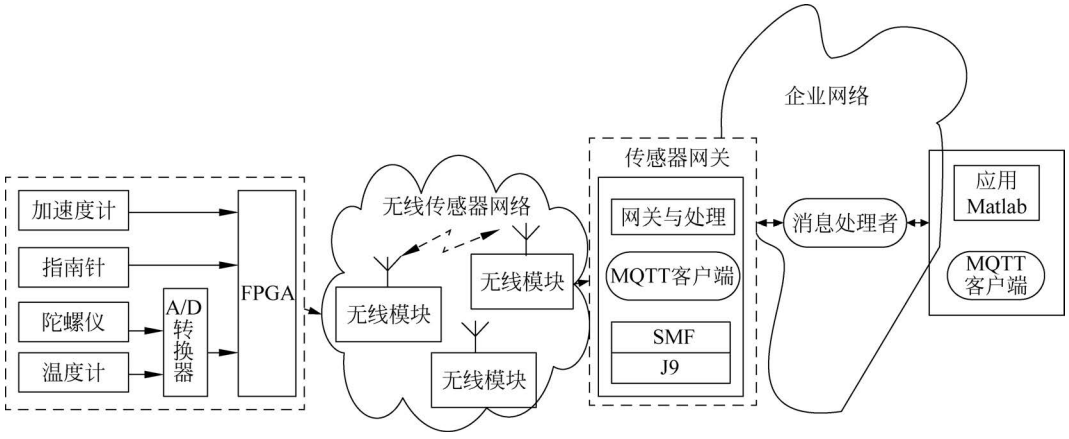


图 3-39　IBM 无线传感器网络测试床

它包括以下几部分。

(1) 配有多种类型传感器的传感器单元。

(2) 一个无线传感器网络。

(3) 一个连接无线传感器网络及企业网络的传感器网关。

(4) 分发传感器数据到各个传感器应用的中间件组件。

(5) 传感器应用软件。

1) 传感器单元

为了可靠地测量特定位置下的方向及动作,传感器单元包括多种类型的传感器,如指南针、加速度计、陀螺仪、温度计。另外,传感器单元还包括一个 A/D 转换器、一块 FPGA 及一

个 RS-232 串行接口。

FPGA 用于收集传感器所产生的数据，将数据组合成数据帧，通过串行接口发送数据帧，串行接口用于连接传感器单元和无线模块。

2）无线传感器网络

无线传感器网络用于将传感器单元的传感器数据帧通过无线接口传送到传感器网关。传感器单元及网关都是通过串行接口与无线模块相连的。无线模块之间以自组织方式形成网状网络。测试过的无线模块有蓝牙模块、IEEE 802.11b 模块、Ember 公司的 IEEE 802.15.4/ZigBee 开发套件、克尔斯博公司的支持 IEEE 802.15.4 的 MicaZ 节点。

3）传感器网关

传感器网关在一个嵌入式 Linux 应用平台上实现，它采用 PowerPC 405 处理器和 MontaVista 的嵌入式 Linux CEE 3.0 操作系统，该传感器网关的结构如图 3-40 所示。

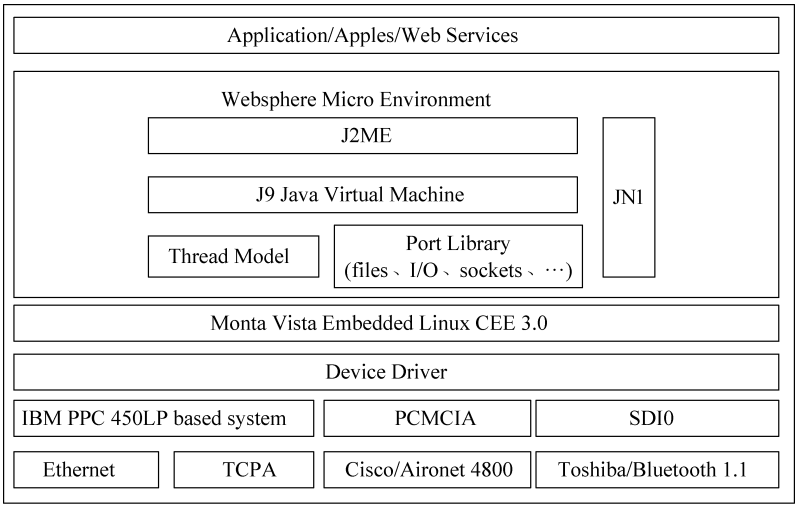


图 3-40　传感器网关的结构

传感器网关作为无线传感器网络与企业网络之间的桥梁，通过 RS-232 接口与无线传感器网络的无线模块相连，通过以太网接口或者无线局域网卡与企业网络相连。

4）中间件

传感器网关使用 MQTT 协议（Message Queuing Telemetry Transport）与企业网络中的传感器应用通信。MQTT 协议需要通过消息处理者（Message Broker）进行消息处理。

另外，中间件还提供了一个服务及网络设备应用部署和维护的平台，采用 SMF（Service Management Framework）来启动、停止和管理消息过滤以及在传感器网关上运行的软件包，且不会影响设备的运行。

5）应用

该测试床用于开发类似于远程测量、移动设备位置感知的应用，采用基于 Matlab 的应用程序将传感器数据图形化，并用 MQTT 协议与消息处理者（Message Broker）通信，获得传感器数据。

其他测试床用于医疗、车辆检测、环境智能等，例如，用于采集信息的移动传感器网络以及 TrueMobile 等。

习题 3

1. 无线 ZigBee 传感器网络系统主要由计算机、_____和 _____ 等组成。

2. 传感器节点的设计主要有 5 点：①微型化，②_____，③_____，④稳定性和安全性，⑤扩展性和灵活性。

3. 大多数传感器网络节点具有终端探测和路由的_____：一方面实现数据的采集和处理；另一方面实现数据的_____。

4. 传感器节点一般由数据处理器模块、_____、_____、传感模块和电源模块 5 部分组成。

5. 电池可分为_____和_____，充电电池能够实现能量补充，电池的内阻较小。

6. TinyOS 是一个开源的_____，它由加州大学伯利克分校开发，主要应用于无线传感器网络方面。它是一种基于组件(Component-Based)的架构方式，能够快速地实现各种应用。TinyOS 程序采用的是_____，程序核心往往都很小。

7. TinyOS 操作系统、库程序和应用服务程序均是用_____。

8. TinyOS 中的组件通常可以分为以下三类：_____、_____、高层次的软件组件。

9. TinyOS 中的消息通信遵循主动消息通信模型，它是一个简单的、_____、面向消息通信(Messaged-Based Communication)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。

10. 简述 WSN 平台硬件设计的主要内容。

11. 简述节点设计的内容。

12. 简述 TinyOS 操作系统的主要特点。

13. 简述平台网关的种类，并指出相应的技术内容及实现的意义。

14. 简述 CC2530 芯片的特点。

15. 简述传感器网络中路由节点的作用。