第3章

物联网无线节点

物联网应用广泛，物联网设备之间也存在着较大差异。从无源 RFID 标签，到电池供电的无线传感器节点，再到有线供电的高清视频监控设备等，无不涵盖在物联网的概念之内。本书以物联网中常用的无线传感器节点为目标设备，讲述物联网中的无线通信技术，并以此为基础展开实践。这些有源的物联网无线节点通过无线方式传输数据，通常不具备有线供电的条件，因此常采用自带电池、能量收集(energy harvesting)等方式供电。

“磨刀不误砍柴工”，无线通信是一个注重工程实践的技术领域。在学习具体的物联网无线通信技术之前，有必要先了解实现这些技术的常用设备，即物联网无线节点，为后续的技术理论学习与实验奠定基础。

本章主要讲述物联网无线节点的硬件系统组成，并初步分析影响无线节点功耗与能耗的主要因素，给出降低功耗与能耗的基本方法；此外，本章也将介绍可用于物联网无线节点的开发板与操作系统实例，并使用 CC1352R1 开发板演示物联网无线节点。

3.1 硬件系统组成

本书中提及的物联网无线节点是指物联网中具有无线通信等功能的小微型嵌入式系统设备，支持物联网数据采集、处理、无线收发等功能，通常自带电池供电，以便于移动与部署。例如，1.4 节中的 SensorTag 就可以看作一个物联网无线节点。

从硬件系统组成上看，一个物联网无线节点至少由通信子系统、处理子系统、电源子系统构成，通常还包含传感等子系统，如图 3.1 所示。传感子系统是指由多种传感器及模数转换器(Analog-to-Digital Converter，ADC)构成的、具有感知某些物理量的值并将其转换成数字信号功能的一系列芯片、模组及电子元器件的集合。其可通过串行接口等途径把采集到的传感器数据传送给处理子系统。处理子系统可对传感器数据做进一步处理，并通过通信子系统发送数据。处理子系统通常也通过串行接口与通信子系统交互数据。通信子系统则负责通过无线方式发送或接收数据。所有硬件子系统都离不开电源子系统的供电支持。

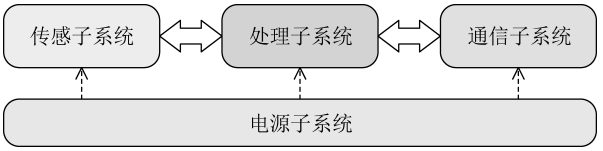


图 3.1 物联网无线节点硬件系统

设计开发一款物联网无线节点需要综合多方面知识，包括传感技术、数字电子技术、无线通信、网络、嵌入式系统、数字信号处理、软件工程等。虽然本书不以设计物联网无线节点为主要目标，但了解物联网无线节点及其系统组成，对于理解后续讲述的技术与理论知识大有裨益。

物联网无线节点，尤其是无线传感器节点，具有无线数据传输功能，并通常自带电池供电，因此具有部署方便灵活、便于移动等优点。随着近年来能量收集技术的发展，使用光、热、风、机械运动、电磁场、磁场、电场等能量来源为无线节点供电逐渐成为可能。但现阶段仍有众多物联网无线节点以电池供电为主。由于物联网无线节点的尺寸和体积通常较小，其自带电池的容量受到限制。然而，人们又希望这些电池供电的物联网无线节点的工作寿命尽可能长，长达 5～10 年甚至更长，而很多物联网无线节点由于其部署环境与成本限制，不便更换电池或无必要更换。因此，短期内，能耗仍是设计开发物联网无线节点时的首要考虑因素之一。未来，随着电池技术与供电技术不断发展，能耗终归不应再是首要考虑因素。

3.1.1 传感子系统

虽然传感子系统并不是物联网无线节点的必需组成部分，但是由于感知外界信息仍是物联网的主要任务之一，很多物联网无线节点都具备感知功能，因此有必要了解支持感知功能的传感子系统。物联网无线节点之间无线传输的数据中，有相当一部分是由传感子系统产生的传感器数据。

由于物联网的应用领域千差万别，其中使用的传感器也有较大差别。传感器(sensor)是能把被测量转换为电信号的器件或装置。被测量包括温度、湿度、压力、烟雾、气体、光线、声音、振动、气流、水流、速度、加速度、距离、位置、海拔高度等(详见 5.1 节)。同一类型传感器的准确度与精度，可能因不同生产商或不同型号而有所不同。有些传感器可在水下的发动机中使用，有些可嵌入人或动物身体中，有些甚至可以在太空中使用。

在物联网无线传感器节点中，与传感器相对应的部件是执行器(actuator)，用于驱动或控制某些物理设备，例如打开阀门。在工业物联网中，电动机、继电器等设备都是常见的执行器。由于在物联网无线节点中，总体上执行器尚未及传感器常见，本书中未专门把执行器作为物联网无线节点的一个硬件子系统单独列出讲解。

表 3.1 列出了部分物联网中使用的传感器及其用途。

本节以意法半导体公司(STMicroelectronics)生产的两款常用传感器为实例，讲解物联网无线节点中的传感子系统。一款为 LM135 精准温度传感器，用于测量温度；另一款为 H3LIS331DL 低功耗高性能三轴数字加速度传感器，用于测量人或物体的三维加

速度。

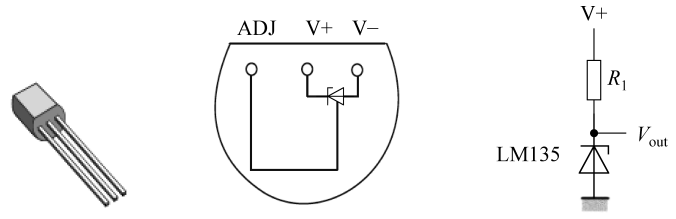
表 3.1　部分物联网中使用的传感器及其用途

传感器	感知的被测量或事件	物联网应用领域
加速度计	人或物体的三维加速度	活火山监测、结构监测、医疗保健、交通、供应链管理等
陀螺仪	人或物体的角速度	医疗保健等
心电图描记器(ECG)	心率	医疗保健
脑电图描记器(EEG)	脑电活动	
肌电图描记器(EMG)	肌肉活动	
血氧计	血氧饱和度	
被动式红外传感器	物体辐射的红外线	医疗保健等
温度传感器	温度	精准农业、医疗保健等
湿度传感器	相对湿度与绝对湿度	精准农业等
电容传感器	溶质浓度	精准农业等
气压计	流体压力	精准农业等
土壤湿度传感器	土壤湿度	精准农业
声发射传感器	裂纹产生的弹性波	结构监测等
声传感器	声压振动	交通、管线监测等
pH 传感器	氢离子浓度	管线监测等
压电圆柱体	气体速度	管线监测等
磁性传感器	磁场的变化	交通等
次声波传感器	地震或火山喷发产生的震荡性声波	活火山监测等
地震传感器	地震波	活火山监测等

LM135 温度传感器的工作温度为－55～150℃，线性输出，误差在 1℃以内。该传感器有 3 个引脚，分别为校准引脚 ADJ、正引脚 V_+、负引脚 V_-，如图 3.2(a)和图 3.2(b)所示。图 3.2(c)为 LM135 的基本应用电路，其中 R_1 应选取使 LM135 中电流为 1mA 的电阻值。如果温度以开尔文(Kelvin，K)为单位，那么 LM135 输出的电压值 V_{out} 正比于被测温度 T，$V_{out}=0.01T$(V)。例如，298.15K(25℃)温度下 LM135 输出的电压值约为 2.9815V。

由于 LM135 的输出为模拟电压值，因此需要先使用 ADC 把模拟信号转换为数字信号，再输入给处理子系统。

H3LIS331DL 加速度传感器支持测量 x、y、z 轴上的 $\pm 100\times 9.8\text{m/s}^2$、$\pm 200\times 9.8\text{m/s}^2$ 或 $\pm 400\times 9.8\text{m/s}^2$ 范围内的加速度值，测量速率为 0.5～1000Hz；支持 SPI(Serial Peripheral Interface)和 I^2C(Inter-Integrated Circuit)串行接口，因此可与处理子系统通过串行接口直接相连。

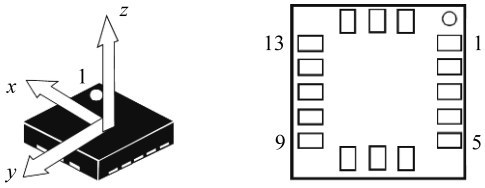


(a) TO–92封装　　(b) 引脚　　(c) 基本应用电路

图 3.2　LM135 温度传感器

（原图来源：LM135 数据手册）

H3LIS331DL 共有 16 个引脚，如图 3.3 所示，各引脚的功能描述见表 3.2。其参考应用电路如图 3.4 所示。综合表 3.2 和图 3.4 可以看出，H3LIS331DL 传感器芯片的外围电路相对简单，其引脚除了电源和接地引脚之外，只有 SPI 或 I^2C 串行接口以及两个中断输出引脚 INT 1 和 INT 2。这两个中断引脚用于在 x、y、z 轴的加速度测量值满足一定条件时，输出中断信号。具体条件：当以 x、y、z 轴的测量值为坐标的三维空间中的点落在被 $x=x_0$、$y=y_0$、$z=z_0$ 三个平面所分割的三维空间的某一区域时，触发中断，其中 x_0、y_0、z_0 分别为 x、y、z 轴上的预设门限值。通过设置 H3LIS331DL 的一些内部寄存器，可灵活设置中断触发条件。除了与中断相关的寄存器，H3LIS331DL 内部还有控制寄存器、状态寄存器、加速度值寄存器等。这些寄存器的读取与写入，包括 x、y、z 轴加速度测量值的读取，都是通过 I^2C 或 SPI 串行接口完成的。



(a) 封装　　(b) 引脚

图 3.3　H3LIS331DL 加速度传感器

（图片来源：H3LIS331DL 数据手册）

表 3.2　H3LIS331DL 的引脚

引　脚　号	名　称	功　能
1	Vdd_IO	I/O 引脚电源
2、3	NC	无连接
4	SCL/SPC	I^2C 或 SPI 的串行时钟
5、12、13、16	GND	0V
6	SDA/SDI	I^2C 的串行数据，或 SPI 的串行数据输入
7	SDO/SA0	SPI 的串行数据输出，或 I^2C 设备地址的最低位
8	CS	SPI 使能，或 I^2C/SPI 模式选择（接 Vdd_IO 选择 I^2C 模式）
9	INT 2	中断 2
10、15	Reserved	保留
11	INT 1	中断 1
14	Vdd	电源

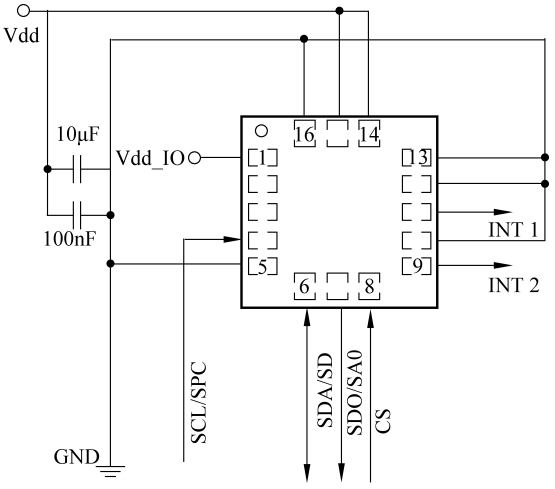


图 3.4　H3LIS331DL 的参考应用电路

（图片来源：H3LIS331DL 数据手册）

3.1.2　SPI 与 I²C 串行接口

物联网无线节点的尺寸和体积通常较小，这就限制了其硬件印制电路板（Printed Circuit Board，PCB）的尺寸，再加上出于制板费用的考虑，通常不会使用较高层数的电路板，这导致了其 PCB 的布线空间较为紧张，限制了芯片之间的走线数量。相比并行接口动辄需要数十条数据线、地址线、控制线，串行接口只需要几条走线。因此，物联网无线节点通常使用 I²C、SPI 等串行接口，作为各个子系统之间的数据传输通路。虽然串行接口的数据传输率不及并行接口，但由于物联网无线节点的数据传输率通常不高，使用串行接口也能满足需求。本节将分别通过 SPI 和 I²C 两种串行接口，读写 H3LIS331DL 加速度传感器的内部寄存器，并读取 x、y、z 轴的加速度测量值。本章后续的处理子系统与通信子系统之间的数据接口，通常也使用这两种或其他串行接口。

1. SPI 串行接口

SPI 是摩托罗拉公司开发的一种用于短距离数据传输的同步串行接口。如图 3.5 所示，主设备通常使用 4 条走线与从设备相连：SCLK（Serial Clock）串行时钟线，用于主设备输出时钟信号给从设备；MOSI（Master Output Slave Input）主设备输出从设备输入数据线，用于传输主设备发送给从设备的数据；MISO（Master Input Slave Output）主设备输入从设备输出数据线，用于传输从设备发送给主设备的数据；CS（Chip Select）片选线，用于主设备选择想要读写的从设备。数据传输由主设备发起，主设备可通过使用多条 CS 片选线的方式，读写多个从设备。图 3.5 中的箭头代表数据传输的方向。

当 H3LIS331DL 的 CS 引脚输入低电平时，该传感器使用 SPI 串行接口传输数据。此时 H3LIS331DL 作为 SPI 从设备工作，其 CS 引脚、SPC 引脚、SDI 引脚、SDO 引脚分别

与主设备的 CS 引脚、SCLK 引脚、MOSI 引脚、MISO 引脚相连。如图 3.6 所示，在主设备读写 H3LIS331DL 的内部寄存器时，主设备先把片选线 CS 拉低，并通过 SPC 时钟线输入时钟信号给 H3LIS331DL，再通过 SDI 数据线告知 H3LIS331DL 将对其哪个寄存器（通过 6 比特的寄存器地址 AD5～AD0 给出）进行何种操作（$R\overline{W}$ 位为 0 时表示写入，为 1 时表示读取；$M\overline{S}$ 位为 0 时读写地址不变，为 1 时自动增加）。若为写入操作，则主设备在 SDI 数据线上后续发送的 DI7～DI0 共 8 比特将被写入地址为 AD5～AD0 的内部寄存器中；若为读取操作，则 H3LIS331DL 将在后续的 8 个时钟周期内通过 SDO 数据线发送地址为 AD5～AD0 的内部寄存器的值 DO7～DO0 共 8 比特，给主设备。

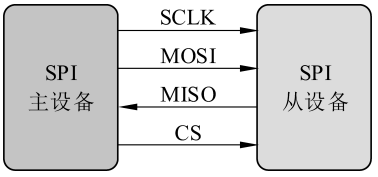


图 3.5 SPI 串行接口

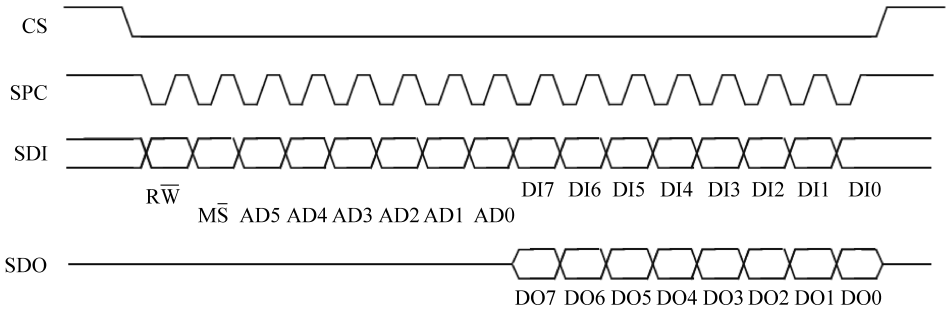


图 3.6 H3LIS331DL 的 SPI 读写协议

（图片来源：H3LIS331DL 数据手册）

【例 3.1】 主设备通过 SPI 串行接口读取 H3LIS331DL 加速度传感器 x 轴加速度的低 8 位数值。

【分析】 通过查阅 H3LIS331DL 数据手册第 30 页以及第 23 页，可知其 x 轴加速度低 8 位数值存放在内部只读寄存器 OUT_X_L 中，该寄存器的 6 比特地址为 101000B。读取内部寄存器时，$R\overline{W}$ 位应置 1。由于只读取当前地址起始的 1 字节数据，$M\overline{S}$ 位可置为 0。因此，主设备除了拉低 H3LIS331DL 的 CS 片选线、在 SPC 时钟线输入时钟信号之外，还需在 SDI 数据线上写入 1 字节的控制位和地址数据 10101000B，之后，通过 SDO 数据线读取 1 字节的 x 轴加速度低 8 位数值。

2. I^2C 串行接口

H3LIS331DL 也支持通过 I^2C 串行接口读写其内部寄存器。I^2C 是飞利浦半导体公司开发的一种支持多个主设备和多个从设备的同步串行接口。如图 3.7 所示，I^2C 的主设备与从设备之间只需用两条双向走线连接：SCL（Serial Clock）串行时钟线，用于主设备发送时钟信号给从设备；SDA（Serial Data）串行数据线，用于主从设备之间的串行数据传输。设备之间使用双向走线相连，设备的主从角色可切换。

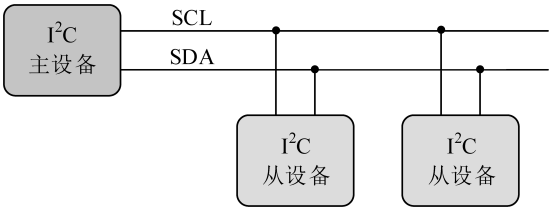


图 3.7　I²C 串行接口

图 3.8 为用 I²C 传输数据的时序图。每次数据传输都是以主设备拉低 SDA 数据线开始的，接着主设备发送时钟信号到 SCL 串行时钟线，然后主设备和从设备按照图 3.9 所示的数据消息格式在 SDA 数据线上逐比特发送或接收数据，直至完成数据传输。最后，主设备停止发送时钟信号，再把 SDA 数据线由低电平置为高电平，让出 SDA 数据线和 SCL 串行时钟线。

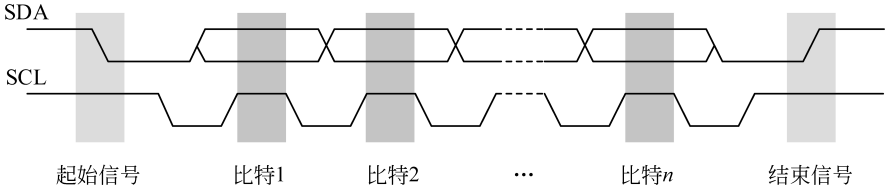


图 3.8　I²C 时序图

由于 I²C 的两条走线上可以连接多个设备，为了区分各个从设备，I²C 支持一个 7 比特长的设备地址(可扩展为 10 比特)。如图 3.9 所示，在数据传输开始后，主设备首先发送一个其想要读写的从设备的地址，并用随后发送的读写位表明主设备想要写入数据给从设备(读写位置 0)还是想要读取从设备数据(读写位置 1)。接着从设备通过在随后的确认位期间拉低 SDA 数据线来应答主设备。然后数据发送端(主设备或从设备)开始发送 1 字节数据，接收端(从设备或主设备)通过确认位应答发送端，直至完成数据传输。

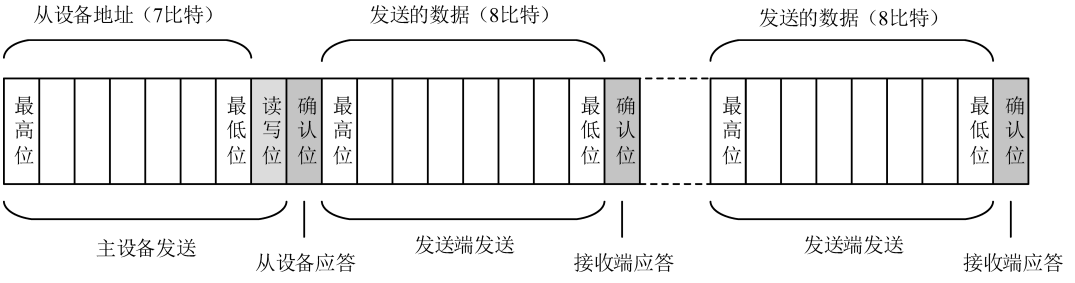


图 3.9　I²C 数据消息格式

当 H3LIS331DL 的 CS 引脚接高电平(Vdd_IO)时，该传感器使用 I²C 串行接口传输数据。此时 H3LIS331DL 作为 I²C 从设备工作，其 SCL 引脚、SDA 引脚分别与主设备的 SCL 引脚、SDA 引脚相连。H3LIS331DL 的 7 比特从设备地址的高 6 比特为 001100B，该 7 比特地址的最低位由 SA0 引脚的输入电平决定：如果 SA0 输入高电平，则 H3LIS331DL 的

从设备地址为 0011001B；如果 SA0 输入低电平，则 H3LIS331DL 的从设备地址为 0011000B。这种做法的好处是，可以把两片 H3LIS331DL 连在同一组 I^2C 走线上，每片独享一个不同的地址。H3LIS331DL 约定每次数据传输的第一个数据字节为子地址字节，该字节的低 7 比特为主设备想要读写的 H3LIS331DL 内部寄存器的地址 AD6～AD0，该字节的最高位为 $\overline{MS}$ 位。当 $\overline{MS}$ 位为 1 时，寄存器地址 AD6～AD0 在多字节读写时自动增加。

【例 3.2】 主设备通过 I^2C 串行接口设置 H3LIS331DL 加速度传感器每秒输出 1000 次 x、y、z 轴上的加速度值。

【分析】 通过查阅 H3LIS331DL 数据手册第 23～25 页，可知所需功能对应的设置位在其内部寄存器 CTRL_REG1 中，该寄存器的 7 比特地址为 0100000B，设置所需功能需把该寄存器改写为 00111111B。写入内部寄存器时，$\overline{RW}$ 位应置 0。由于只写入当前地址起始的 1 字节数据，$\overline{MS}$ 位可置 0。因此，主设备通过 I^2C 串行接口应发送的第一个字节单元是 00110000B（假设 SA0 引脚输入为低电平），第二个字节单元是 00100000B，第三个字节单元是 00111111B。

综上，处理子系统可以通过 SPI 或 I^2C 串行接口读写 H3LIS331DL 加速度传感器的内部寄存器。这两种串行接口的不同：I^2C 串行接口只需要两条走线，通过地址来寻址从设备；而 SPI 串行接口一般使用 4 条走线，由于有 CS 片选线，无须寻址从设备；当 I^2C 串行接口连接多个主设备时，可能会产生发送数据冲突，而 SPI 的 4 条走线中只有一个主设备，不会产生冲突；在数据传输率方面，SPI 接口的传输率只取决于主从设备，而标准化的 I^2C 接口的传输率则受标准限制：标准模式下为 100kb/s，快速模式下为 400kb/s，高速模式下为 3.4Mb/s，超高速模式下为 5Mb/s。

3.1.3 处理子系统

传感子系统采集数据，处理子系统处理数据。物联网无线节点可以直接对采集的数据进行处理，也可以将数据发送到其他设备（例如，汇聚节点、网关、基站、服务器等）进行进一步处理、分析、汇聚，这取决于节点的处理能力、电池剩余电量、通信协议等因素。此外，部分通信协议的实现也常由处理子系统完成。

处理子系统以处理器为中心。物联网无线节点由于受尺寸、成本、电池容量等因素制约，常使用微控制器（MicroController Unit，MCU）或数字信号处理器（Digital Signal Processor，DSP）作为其主要处理器。用户可在这两类处理器上运行自己的程序来实现所需功能，包括实现算法、协议等。8 位 MCU 的批量售价可低至几角每片，32 位 MCU 的批量售价可低至几元每片。这两类处理器可以满足众多应用的需求。在一些需要定制化处理器的应用领域，可使用现场可编程门阵列（Field-Programmable Gate Array，FPGA）或专用集成电路（Application-Specific Integrated Circuit，ASIC）作为主要处理器，但二者的成本相对较高。FPGA 的并行处理能力强，但芯片的售价相对较高，使得产品的量产成本相对较高，并且 FPGA 的功耗相对较高，不大适用于依靠电池供电的物联网无线节点。

ASIC 的量产成本相对较低，能效高，但其非重复性工程（Non-Recurring Engineering，NRE）的费用较高，即开发、测试、流片等工程成本较高，甚至可达数千万元。在需求量持续较大的应用领域，为了进一步降低量产成本与功耗，常使用 ASIC。本节主要介绍设计物联网无线节点时常用的两类处理器：MCU 和 DSP。

（1）MCU 是集成在一片芯片上的小计算机，常用于自动控制和嵌入式系统领域，也被称单片机。MCU 在一片芯片上集成了一个或多个处理器核、易失性存储器、非易失性存储器、时钟发生器，以及串行通信接口、ADC、输入输出位等外围设备。传统的 MCU 多基于冯·诺依曼体系结构，即程序与数据合用一条连接处理器核与存储器的总线。常见的处理器核包括高级精简指令集计算机（Advanced RISC Machine，ARM）核、无内部互锁流水级的微处理器（Microprocessor without Interlocked Pipelined Stages，MIPS）核等；易失性存储器包括静态随机存储器（Static Random-Access Memory，SRAM）、同步动态随机存储器（Synchronous Dynamic Random-Access Memory，SDRAM）等；非易失性存储器包括闪存（flash memory）、电擦除可编程只读存储器（Electrically-Erasable Programmable Read-Only Memory，EEPROM）等；串行通信接口包括 I^2C、SPI 等。传统的 MCU 多为定点型处理器，浮点运算只能靠程序来完成。MCU（尤其是中、低端 MCU）一般适用于数学运算量不大的应用；对于数学运算量较大的应用，可以考虑使用 DSP。

（2）DSP 是专门为数字信号处理而优化了的微处理器。数字信号处理算法，如数字滤波、快速傅里叶变换等，常需要周期性地完成大量数学运算（例如，乘法与加法）及数据读写操作，且这些操作常需要在指定的时间内完成。虽然许多 MCU 也支持运行数字信号处理算法，但常因处理器硬件在数学运算及数据读写等方面性能有限而难以保证在指定时间内完成这些数学运算。此外，使用 DSP 来执行数学运算等任务，能效常高于传统 MCU。DSP 通常基于哈佛体系结构，即数据存储空间与程序存储空间相分离，在处理器核与存储器之间，除了有一条程序总线，还有一条或多条数据总线。DSP 通常是超标量（superscalar）处理器或超长指令字（Very Long Instruction Word，VLIW）处理器，前者通过向处理器内不同执行单元同时分派多条指令以达到在一个时钟周期内执行多条指令的效果，后者在程序编译时就确定了在一个时钟周期内哪些指令可以同时执行。DSP 既有定点型也有浮点型，后者从硬件上支持浮点运算。由于 DSP 的指令集为数学运算做了专门优化，处理器内又有专门的硬件支持，在传统 MCU 上需要经过若干个时钟周期执行若干条指令才能完成的任务，在 DSP 上甚至只需一个时钟周期一条指令即可完成，如乘累加操作。许多传统 MCU 上没有硬件乘法器，但所有的 DSP 上都有硬件乘法器，且通常有多个。此外，DSP 通常还具备硬件模寻址循环缓冲区、硬件控制循环等专门的硬件支持。

作为 MCU 的一个实例，ARM Cortex-M4F 处理器面世于 2010 年前后，是一款具有 DSP 特点的、基于哈佛体系结构的、具有三级流水线的 32 位 MCU。ARM Cortex-M4F 基于 ARMv7-M 架构，在 Cortex-M3 的基础上，增加了 DSP 指令和单精度浮点单元，并带有 32 位的硬件乘法器和 32 位的硬件除法器，支持睡眠模式。

3.1.4 通信子系统

通信子系统负责数据的无线接收与发送，与处理子系统常通过串行接口相连接。在物联网无线节点中，通常使用现成的无线收发器芯片来实现通信子系统、完成数据的无线接收与发送。较早面世的用户可配置的无线收发器芯片有美国 TI 公司的 CC 系列芯片、挪威 Nordic 半导体公司的 nRF 系列芯片等。本节以 TI 公司的 CC1352R 无线微控制器为例，介绍通信子系统。

CC1352R 是 TI 公司 2018 年推出的一款支持多协议（包括 ZigBee、Thread、BLE、IEEE 802.15.4g、6LoWPAN、mioty 等）、多频段（1GHz 以下和 2.4GHz 频段）的无线微控制器芯片，适用于低功耗传感与无线数据传输。CC1352R 内置一个时钟频率为 48MHz 具有 352kB 闪存和 80kB SRAM 的 ARM Cortex-M4F 微控制器，一个具有 4kB SRAM 的超低功耗传感器控制器，一个兼容蓝牙和 IEEE 802.15.4 等标准的支持 1GHz 以下和 2.4GHz 频段的无线收发器，以及丰富的串行接口等外围设备。CC1352R 芯片有 48 个引脚，其部分引脚的功能说明如表 3.3 所示。

表 3.3 CC1352R 的部分引脚（电源、接地、调试引脚除外）

引 脚 号	名 称	功 能
1	RF_P_2_4GHZ	正 2.4GHz 射频输入输出
2	RF_N_2_4GHZ	负 2.4GHz 射频输入输出
3	RF_P_SUB_1GHZ	正 1GHz 以下射频输入输出
4	RF_N_SUB_1GHZ	负 1GHz 以下射频输入输出
5	RX_TX	可选的射频低噪声放大器偏置引脚
6	X32K_Q1	32kHz 晶体振荡器引脚 1
7	X32K_Q2	32kHz 晶体振荡器引脚 2
46	X48M_N	48MHz 晶体振荡器引脚 1
47	X48M_P	48MHz 晶体振荡器引脚 2
35	RESET_N	复位引脚，低电平有效
8～12、14～21、26～32、36～43	DIO	通用输入输出引脚

CC1352R 可用来实现物联网无线节点的通信子系统（因其集成了 RF 核，RF 即 Radio Frequency，译为射频）以及处理子系统（因其集成了 48MHz 的 ARM Cortex-M4F 微控制器）。CC1352R 的 RF 核包含一个 ARM Cortex-M0 微控制器和一个调制解调器、频率合成器及射频接口模块，前者负责接收 ARM Cortex-M4F 微控制器发送来的高级命令并在 RF 核内调度执行，后者是射频的核心部分。

高级命令可分为三类：射频操作命令、立即命令、直接命令。射频操作命令可操作射频硬件完成发送或接收一帧数据、设置射频硬件寄存器或更复杂的依赖于协议的操作。立即命令用于改变或返回射频状态，如接收信号强度，或操作发送或接收的数据队列。直

接命令则是无参数或短参数的立即命令。借助这些高级命令，可实现介质接入控制子层的基本操作，屏蔽了物理层的实现细节，只需通过命令设置物理层的基本参数即可，方便了使用，简化了开发。

【例 3.3】 在 CC1352R 使用专有射频接收数据时，跳过正在接收的数据。

【分析】 通过查阅 CC1352R 技术参考手册第 1905 页、第 2031 页，可知向 RF 核发送命令通过向 CMDR 寄存器中写入命令的方式实现。要跳过当前正在接收的数据，可向 CMDR 写入一条 ID 为 3402H 的直接命令 CMD_PROP_RESTART_RX。直接命令的格式：命令的第 16～31 位为命令 ID，第 0～1 位为 1H，因此可通过向 CMDR 寄存器写入 32 比特值 34020001H 完成所需操作。

CC1352R 并无固定的串行接口引脚，可通过其内部的输入输出（Input/Output，I/O）控制器把串行接口映射到物理引脚 DIO 上，其支持一个 I^2C 接口和两个 SPI 接口。

【例 3.4】 分别将 CC1352R 的 10 号引脚配置为 I^2C 串行接口的时钟线、11 号引脚配置为 I^2C 串行接口的数据线。

【分析】 通过查阅 CC1352R 数据手册第 7 页，可知其 10 号引脚的名称为 DIO_5、11 号引脚的名称为 DIO_6，均为可配置的数据输入输出引脚。再通过查阅 CC1352R 技术参考手册第 1133～1142 页，可知 DIO_5 对应的配置寄存器为 IOCFG5，寄存器地址偏移量为 14H；DIO_6 对应的配置寄存器为 IOCFG6，寄存器地址偏移量为 18H。通过将 IOCFG5 配置寄存器的第 0～5 位 PORT_ID 字段置为 EH、将 IOCFG6 配置寄存器的第 0～5 位 PORT_ID 字段置为 DH，可完成所需配置。如果需要在该引脚上输入数据，还要将对应配置寄存器中的第 29 位 IE 字段的值置为 1。

3.2　功耗与能耗

功耗是指每单位时间内消耗的电能；能耗是指所消耗的电能。能耗目前仍是开发设计由电池供电的物联网无线节点时的首要考虑因素之一。

Raghunathan 等人在 2002 年发表的论文中给出了 MEDUSA-Ⅱ节点的功耗，MEDUSA-Ⅱ是由加州大学洛杉矶分校开发的无线传感器节点。该节点的传感子系统与处理子系统在工作时的总功耗为 9.72mW，此时若开启通信子系统并工作在接收模式下，则总功耗将上升至 22.2mW，即接收模式下通信子系统的功耗大约为 12.48mW，约是传感子系统和处理子系统功耗之和的 1.28 倍。如果通信子系统工作在发送模式下，其功耗通常还会再高一些。由此可见，就该节点而言，通信子系统的功耗相对较大。若功耗较大的通信子系统长时间处于工作状态，则其能耗也较大。那么，影响通信子系统能耗的因素有哪些？

通过简单能耗模型对通信子系统的能耗做初步分析，如图 3.10 所示。通信子系统的

接收机常由混频器、频率合成器、压控振荡器、锁相环、解调器、功率放大器等部件组成，它们在工作时都需要耗电。把接收机的接收电路平均接收每比特数据所消耗的电能记作 e_{rx_elec}。在只考虑接收电路能耗时，接收机接收 k 比特数据所消耗的电能为

$$E_{rx}(k)=E_{rx_elec}(k)=k\cdot e_{rx_elec} \tag{3-1}$$

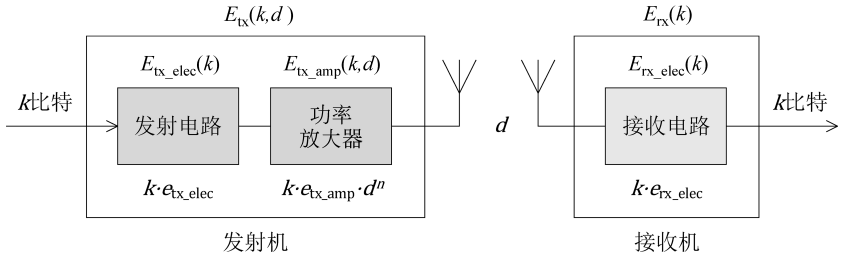


图 3.10　简单能耗模型

发射机和接收机之间的一个主要区别是发射机通常使用输出功率相对较大的功率放大器。同样，把发射机的发射电路平均发送每比特数据所消耗的电能记作 e_{tx_elec}，则发射机发送 k 比特数据所消耗的电能为 $E_{tx_elec}(k)=k\cdot e_{tx_elec}$。由于发射机的输出功率较大，该模型把发送 k 比特数据到 d 米远处时功率放大器所消耗的电能建模为 $E_{tx_amp}(k,d)=k\cdot e_{tx_amp}\cdot d^n$，其中 e_{tx_amp} 为系数，它的值取决于功率放大器的效率、发射天线的增益、载波的波长、热噪声的功率谱密度、所需的信噪比等因素；d 为传输距离；n 为路径损耗指数（详见 4.1 节），通常 n 为 2～6。由此可得，发射机的总能耗为

$$E_{tx}(k,d)=E_{tx_elec}(k)+E_{tx_amp}(k,d)=k\cdot e_{tx_elec}+k\cdot e_{tx_amp}\cdot d^n \tag{3-2}$$

可见，通信子系统工作在接收模式下时，其接收的数据越多，能耗越大；工作在发送模式下时，发送的数据越多，传输距离越远，能耗越大。

电池供电的物联网无线节点在电池容量有限的情况下，若要延长节点的工作寿命，所能采取的措施离不开开源与节流。开源是指用额外的电源或能源为节点供电或充电，如使用额外的电池或利用太阳能、热能、风力、机械运动、电磁波、磁力等能量来源为电池充电等。节流指减少节点自身的能耗，尤其在受成本、尺寸、工作条件等因素制约而不便或无法开源的情况下，节流成了唯一选择。

在物联网无线节点软硬件及无线通信技术的不同功能层面上，都有各自的有助于节省能耗的措施与技术方法。例如，对于通信子系统而言，可选用低功耗、高效的无线收发器芯片，在减小传输距离的同时尽量降低发射机输出功率；对于无线通信物理层而言，可选用适合的调制方式与编码技术，选取较低的载波频率，尽量降低数据传输率；对于介质接入控制子层而言，可选用高效的介质接入控制协议，减少空闲监听、旁听、冲突、盲发、控制帧开销及重传次数；对于网络层而言，可选用适合的网络协议、选择节能的传输路径等。如何在满足工作需要的情况下尽量降低物联网无线节点与网络的整体能耗，需要综合考虑与权衡，是一项较为复杂的系统工程。

在物联网无线节点中，数据的采集、处理、传输，都离不开处理子系统。降低处理子系

统的能耗也有助于降低节点的能耗。那么，处理子系统的功耗和能耗与哪些因素有关？如何降低处理子系统的能耗？

处理子系统以处理器为中心。处理器集成电路多使用互补金属氧化物半导体(Complementary Metal Oxide Semiconductor，CMOS)生产工艺制造。CMOS 电路的功耗 P 为

$$P = P_t \cdot C_L \cdot V \cdot V_{dd} \cdot f_{clk} + I_{sc} \cdot V_{dd} + I_{leakage} \cdot V_{dd} \tag{3-3}$$

式中　P_t——耗电跃迁发生的概率；

C_L——负载电容；

V——电压摆幅，大多数情况下，与供电电压V_{dd}相同；

V_{dd}——供电电压；

f_{clk}——时钟频率；

I_{sc}——直流短路电流；

$I_{leakage}$——漏电流。

式(3-3)为三项乘积之和。其中，第一项为开关电流产生的动态功耗；第二项为直流短路电流产生的功耗；第三项为漏电流产生的静态功耗。由此可见，通过降低 CMOS 电路的供电电压 V_{dd}或时钟频率f_{clk}可以降低功耗。

人们常采用功率管理等技术来降低微处理器的功耗，包括动态电压与频率调节、电源门控、时钟门控等技术。

动态电压调节(dynamic voltage scaling)是指根据情况提高或降低微处理器芯片中使用的电压。动态频率调节(dynamic frequency scaling)是指根据实际需要自动调整微处理器的时钟频率。这两种方法的目的是在微处理器处于工作状态时调整其性能。降低电压与时钟频率都可以降低功耗并减少芯片产生的热量。既然如此，是否可以持续降低微处理器的电压？

实际上，微处理器的时钟频率 f_{clk}与供电电压V_{dd}之间存在如下关系：

$$f_{clk} \leqslant \frac{K \cdot (V_{dd} - V_{th})^a}{V_{dd}} \approx K \cdot (V_{dd} - c) \tag{3-4}$$

式中，不同的微处理器的 a、K、c、V_{th}值不同。由式(3-4)可知，对于每个微处理器的每个时钟频率值，都有一个最小的供电电压值。将供电电压降低到这个值是降低功耗而不影响性能的有效方法。

电源门控(power gating)是指通过切断未使用的电路模块的电流来降低其功耗。时钟门控(clock gating)在同步电路中较为常见，其通过在未使用电路模块时移除其时钟信号来降低动态功耗。借助于时钟门控等技术，可以通过配置微处理器运行在不同功耗模式下，来降低其总体功耗及能耗。例如，若让 MEDUSA-Ⅱ节点运行在睡眠模式下，其功耗将降低至 0.02mW，不到运行在接收模式下功耗的 1/1000。

不同微处理器支持的功耗模式不尽相同。这里以 TI 公司的 MSP430x4xx 系列超低功耗微处理器为例，说明微处理器的功耗模式。该系列处理器支持 6 种功耗模式，分别为活跃模式(Active Mode，AM)与低功耗模式(Low-Power Mode，LPM)，低功耗模式包括：LPM0、LPM1、LPM2、LPM3、LPM4。在不同功耗模式下，处理器及时钟信号的状态不

同，如表 3.4 所示。

表 3.4 MSP430x4xx 的功耗模式

功耗模式	处理器	主时钟	数控振荡器	子系统主时钟	辅助时钟
AM	开启	开启	开启	开启	开启
LPM0	关闭	关闭	开启	开启	开启
LPM1	关闭	关闭	仅发生器开启	开启	开启
LPM2	关闭	关闭	仅发生器开启	关闭	开启
LPM3	关闭	关闭	关闭	关闭	开启
LPM4	关闭	关闭	关闭	关闭	关闭

如图 3.11 所示，MSP430x4xx 在低功耗模式 LPM4 下的电流消耗可低至 AM 模式下的 1/3000。因此，让微处理器尽可能长时间地工作在尽可能低的功耗模式下，将会节省大量能耗。

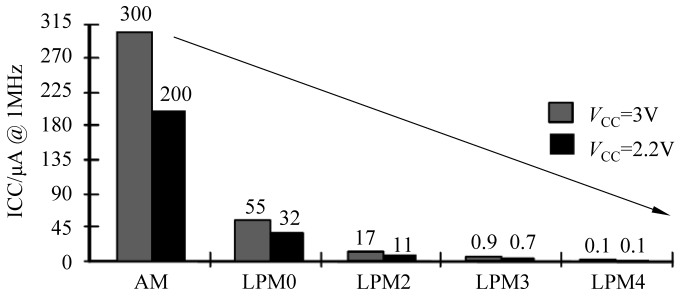


图 3.11 MSP430x4xx 不同功耗模式下的电流消耗

（图片来源：MSP430x4xx 用户指南）

3.3 开发板与操作系统

物联网无线节点常由于内部存储空间、计算能力、电池容量等资源有限而功能受限。从现阶段存储与处理能力的角度，可以把物联网设备大致分为低端、中端、高端三类。

- 低端设备是资源严重受限的设备，无法运行传统的操作系统（例如，Linux、Windows IoT 版）。其随机存储器（Random-Access Memory，RAM）约几千字节、闪存约几十千字节，处理器的时钟频率为几兆赫兹到几十兆赫兹。低端设备通常需要借助网关等设备方能接入互联网。低端设备主要用来执行基本的感知与控制任务，可用于实现无线传感器网络中的终端节点。
- 中端设备是资源较少的设备，具备一些处理能力，但不足以满足非常复杂的需求。其 RAM 约几十千字节、闪存约几百千字节，处理器的时钟频率为几十兆赫兹到几百兆赫兹。中端设备可使用专门为资源受限节点设计的协议栈接入互联网，它们可用来实现物联网系统中的网关等设备。网关是连接两个不同通信网络的设备。

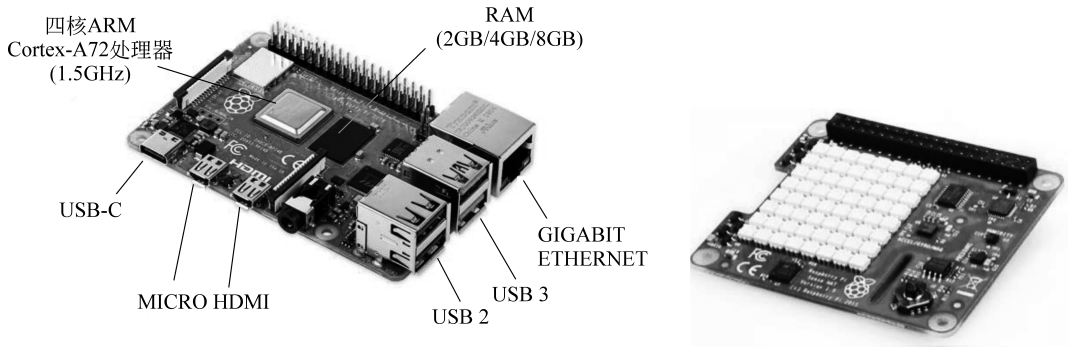
- 高端设备是具有足够资源的设备，包括功能强大的处理器、大量的 RAM 与闪存，可运行传统的操作系统以及机器学习等算法，支持几乎所有的通信协议。它们可用来实现物联网系统中的复杂网关等设备。

物联网无线节点的处理子系统可由上述三类设备实现，根据具体的应用与功能需求，并结合传感子系统与通信子系统的需求，选择适合的设备。在设计与实现物联网无线节点时，可以上述设备的开发板和适用于这些设备的操作系统为起点进行开发，以减少工作量、缩短开发周期。当然，对于需求量较大或者对成本、尺寸、功耗等方面要求较高的应用，也可以从设计物联网无线节点的 PCB 开始。

作为实例，本节介绍两款常见的可用于开发物联网无线节点的开发板以及适用于物联网设备的 5 款常见操作系统。开发板(development board)是指可用于某些应用系统软硬件开发与测试的 PCB，板上包含以某些芯片或元器件(例如，微控制器芯片)为中心的一系列硬件组件。

市面上可用于开发物联网无线节点的开发板很多，它们在性能、售价、尺寸、功耗、供电方式、使用的无线通信技术等方面的差别也较大。目前常见的高、中端开发板包括树莓派(Raspberry Pi)系列和 Arduino 系列开发板等。

- 树莓派是英国树莓派基金会开发的一系列小型单板计算机，它们的售价较低、尺寸较小，在相关领域的研究与开发项目中被广泛使用。2019 年发布的树莓派 4 开发板如图 3.12(a)所示，售价低至 35 美元，配有主频为 1.5GHz 的 64 位四核 ARM Cortex-A72 处理器、2GB/4GB/8GB 的 RAM 及丰富的接口，支持 2.4GHz 和 5GHz 频段的 WiFi、蓝牙 5.0、BLE 无线通信，但功耗相对较高，至少需要 15W 的电源供电。Sense HAT 是一块树莓派 4 的附加板，可插在树莓派 4 开发板上，如图 3.12(b)所示。Sense HAT 具有一个 8×8 的彩色 LED 矩阵、一个有 5 个按钮的操纵杆，并包括陀螺仪、加速度计、磁力计、温度传感器、气压传感器、湿度传感器等。可以用树莓派 4 和 Sense HAT 实现处理能力较强的高端物联网无线节点。



(a) 树莓派 4 开发板　　(b) Sense HAT

图 3.12　树莓派 4 与 Sense HAT

(原图来源：Raspberry Pi Foundation)

- Arduino 是一系列易于使用的硬件模块与开发板。其中一款名为 Arduino Nano 33 IoT 的开发板如图 3.13 所示，售价约 21 美元，配有主频为 48MHz 的 32 位 ARM Cortex-M0 处理器、32kB 的 SRAM、256kB 闪存、一个 SPI 接口和一个 I^2C 接口，并支持 2.4GHz 频段的 WiFi、蓝牙及 BLE 无线连接。此外，该开发板还配有加速度计和陀螺仪，使得该开发板适用于振动报警系统、相对定位、计步器等应用场景。可以用 Arduino Nano 33 IoT 实现相对简单的中、低端物联网无线节点。

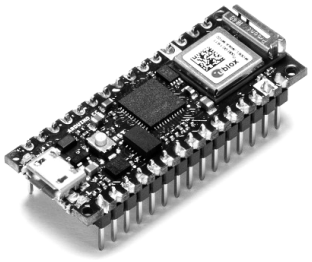


图 3.13 Arduino Nano 33 IoT

（图片来源：Arduino）

尽管使用上述开发板支持的 WiFi、蓝牙和 BLE 无线连接可以满足一些应用的需求，但尚有一些需求无法满足，如超低功耗、远距离无线传输、节点众多的自组网、1GHz 以下频段等。若使用其他现有的无线通信标准可以满足这些需求，可优先考虑使用支持该现有无线通信标准的开发板实现物联网无线节点。若尚无现有的无线通信标准可满足特定的应用需求，也可以基于本书后续讲解的无线通信技术及现有的无线收发器（例如 CC1352R），自行设计并实现可满足这些特定应用需求的无线通信系统与物联网无线节点。

在物联网无线节点的软件开发方面，使用操作系统可能有助于简化应用软件的开发。但并非每个物联网无线节点都需要操作系统才能工作。如果应用软件比较复杂并且运行在高端设备上，则通常使用操作系统。如果应用软件比较简单并且运行在低端设备上，则通常不需要操作系统。在对程序效率和微处理器功耗有较高要求时，也可以不使用操作系统。

在物联网无线节点中，常见的操作系统除了 Linux 和 Windows IoT 版外，还包括 FreeRTOS、Mbed OS、Contiki、TinyOS、RIOT 等。在开发物联网无线节点时，可根据实际需要选用适合的操作系统。

- FreeRTOS 是一个轻量级的实时操作系统，包括一个内核和一组库。其内核本身仅包含 3 个 C 文件。FreeRTOS 提供了用于多线程多任务、互斥锁、信号量、软件定时器的方法，并支持线程优先级。
- Mbed OS 是一个基于 ARM Cortex-M 微控制器的专门为物联网设计的嵌入式操作系统。它由核心库组成，这些库提供了开发基于 ARM Cortex-M 微控制器的物联网节点所需的功能，包括安全、连接、实时操作系统，以及用于传感器和输入输出设备的驱动程序。
- Contiki 是一个用于内存受限的网络设备的操作系统，尤其适用于低功耗物联网无线设备。Contiki 支持多任务处理，并内置 TCP（Transmission Control Protocol，传输控制协议）/IP 协议栈，只需要约 10kB 的 RAM 和 30kB 的 ROM（Read-Only Memory，只读存储器）。包括图形用户界面的完整系统需要大约 30kB 的 RAM。Contiki 的一个新分支为 Contiki-NG，是用于物联网中资源受限设备的操作系统。Contiki-NG 包含低功耗 IPv6（Internet Protocol version 6，互联网协议第 6 版）协议栈，可在 ARM Cortex-M3/M4 和 TI MSP430 等低功耗微控制器上运行，约需

要 100kB 的 ROM，至少 10kB 的 RAM。

- TinyOS 是一个专为低功耗无线设备而设计的操作系统。它用 nesC 编程语言编写，nesC 是针对内存受限传感器网络而优化的 C 语言扩展。TinyOS 程序由软件组件构建，组件使用接口相互连接。TinyOS 提供了用于分组通信、路由、传感、驱动及存储的接口和组件。但其从 2012 年以来没有更新过。
- RIOT 是一种用于内存受限的网络设备的小型操作系统，支持大多数低功耗物联网设备和微控制器架构。RIOT 基于微内核架构，允许应用程序使用 C 和 C++ 语言进行编程，并提供完整的多线程和实时功能。RIOT 提供多个网络协议栈，包括 IPv6、6LoWPAN（IPv6 over Low-power Wireless Personal Area Networks，低功耗无线个域网上的 IPv6）、RPL（Routing Protocol for Low-power and lossy networks，低功耗有损网络的路由协议）、UDP（User Datagram Protocol，用户数据报协议）、TCP、CoAP（Constrained Application Protocol，受限应用协议）等。

3.4　本章实验

本书选用基于 CC1352R 芯片的开发板 LAUNCHXL-CC1352R1（在本书中简称 CC1352R1），作为物联网无线节点的实验平台。

CC1352R1 开发板如图 3.14 所示。该开发板上除了 CC1352R 芯片，还带有 XDS110 仿真器，只需用一根 USB（Universal Serial Bus，通用串行总线）线缆连接该开发板与计算机，即可通过计算机对该开发板进行编程与调试。此外，该开发板带有一个 40 针的 BoosterPack 扩展接口，可外接 TI 公司的 430BOOST-SHARP96（简称 SHARP96）、BOOSTXL-SHARP128（简称 SHARP128）等液晶显示模块以及其他功能模块，便于开发与功能扩展。在本书后续的一些实验中，需要同时使用至少两块 CC1352R1 开发板，每块开发板可以选配一个 SHARP128 或 SHARP96 液晶显示模块。

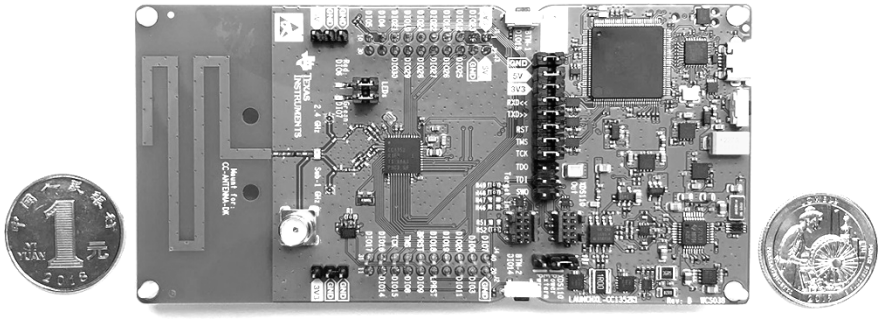


图 3.14　LAUNCHXL-CC1352R1 开发板

【实验 3.1】　使用蓝牙无线连接的物联网无线节点。

作为使用 CC1352R1 开发板的第一个实验，将在该开发板上编译运行一个使用蓝牙无线连接的物联网无线节点演示程序，以便熟悉该开发板的集成开发环境、掌握编译运行程序的步骤、进一步理解物联网无线通信的概念。

(1) 在支持蓝牙与 WiFi 连接的智能手机(或平板计算机)上安装名为 SimpleLink Starter 的 App,可通过 iOS 应用商店或安卓应用市场安装。

(2) 在计算机上安装 CCS(Code Composer Studio)集成开发环境和 SimpleLink CC13x2 软件开发工具包。

(3) 使用 CC1352R1 开发板附带的 USB 线缆,通过 USB 接口将开发板连接到计算机上。

(4) 在计算机上运行 CCS 集成开发环境。CCS 启动后的界面如图 3.15 所示。单击打开 Resource Explorer。

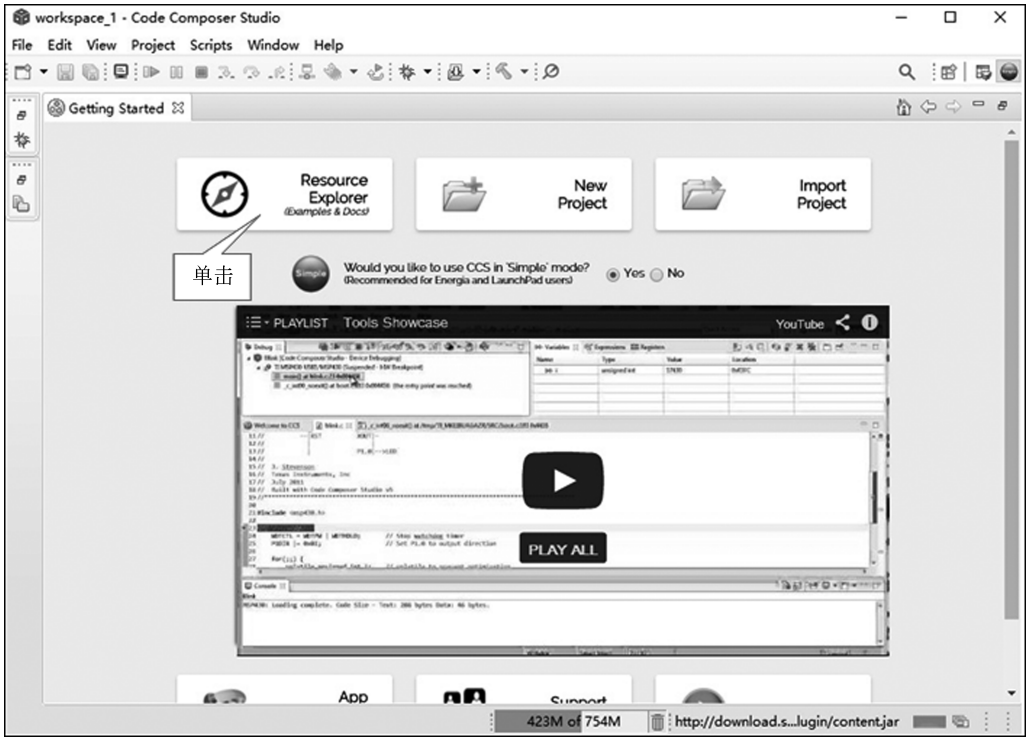


图 3.15 CCS 启动后的界面

(5) 导入演示项目。在 TI Resource Explorer 左侧的导航栏中,依次选择 Software→SimpleLink CC13x2 26x2 SDK→Examples→Development Tools→CC1352R LaunchPad→BLE5-Stack→project_zero→TI-RTOS→CCS Compiler 选项,在单击 project_zero 后,单击右侧的 Import 按钮导入项目,如图 3.16 所示。

(6) 构建项目。单击 Project 菜单下的 Build All 命令,如图 3.17 所示。

(7) 下载程序到开发板准备运行。单击如图 3.18 所示的下三角按钮打开调试菜单,再单击 Code Composer Debug Session 命令。也可以直接单击图 3.18 中下三角按钮左侧的图标。

(8) 打开串口终端窗口,用于查看即将在开发板上运行的程序的串口输出。单击如图 3.19 所示的 Open a Terminal 图标,在弹出的窗口中选择 Serial Terminal,并选择串口。

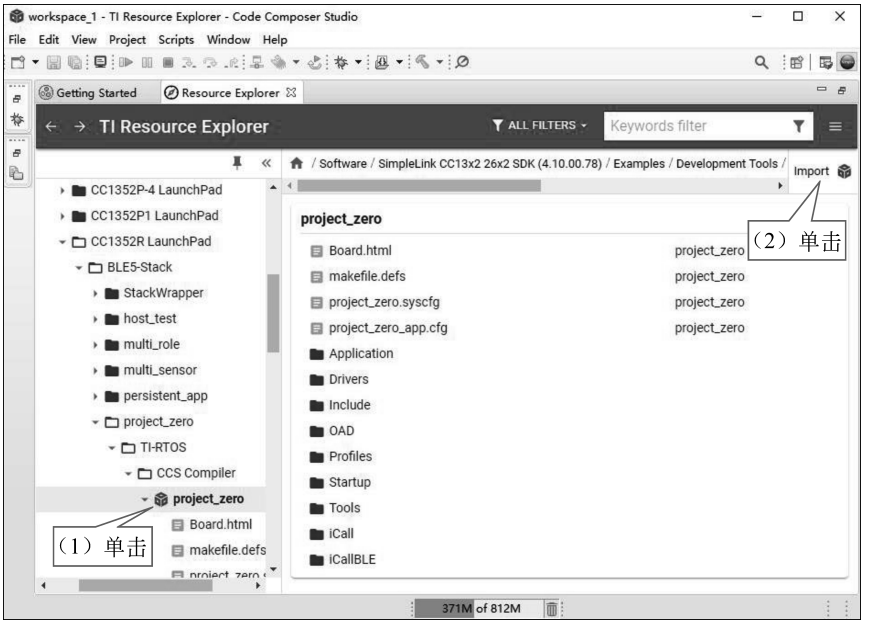


图 3.16　导入项目

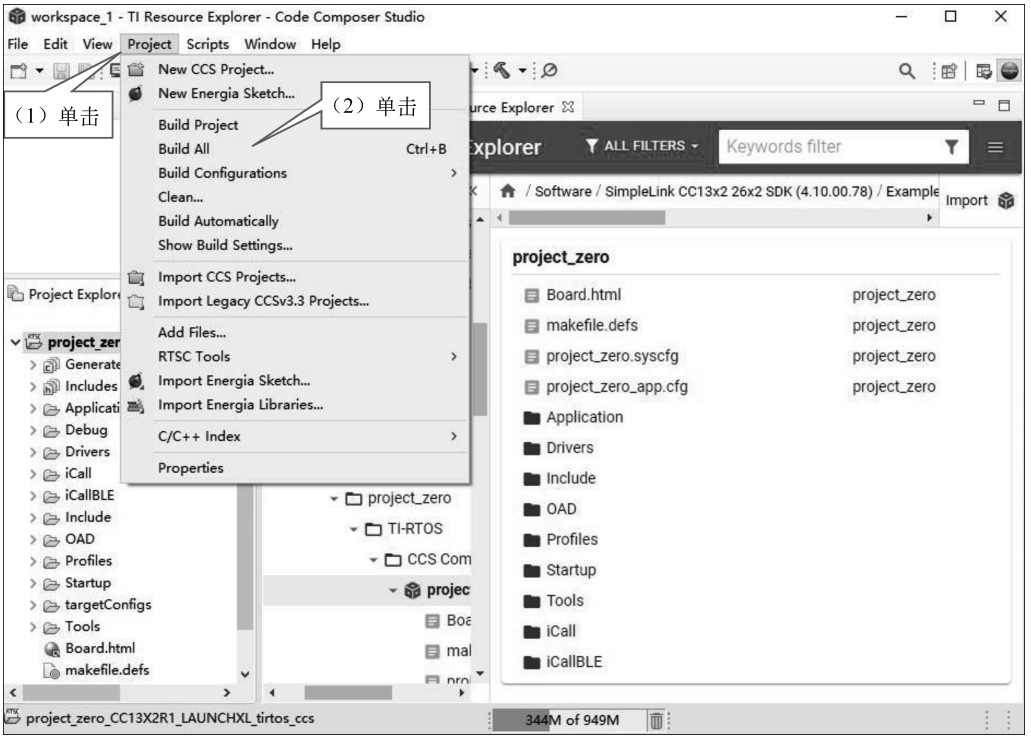


图 3.17　构建项目

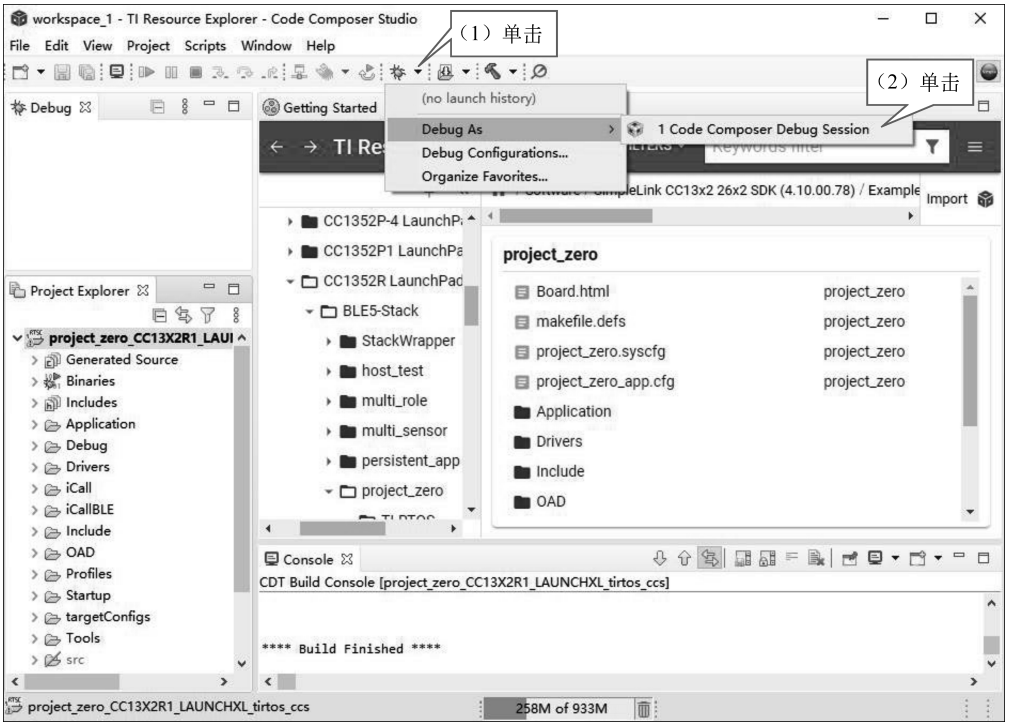


图 3.18　下载程序到开发板

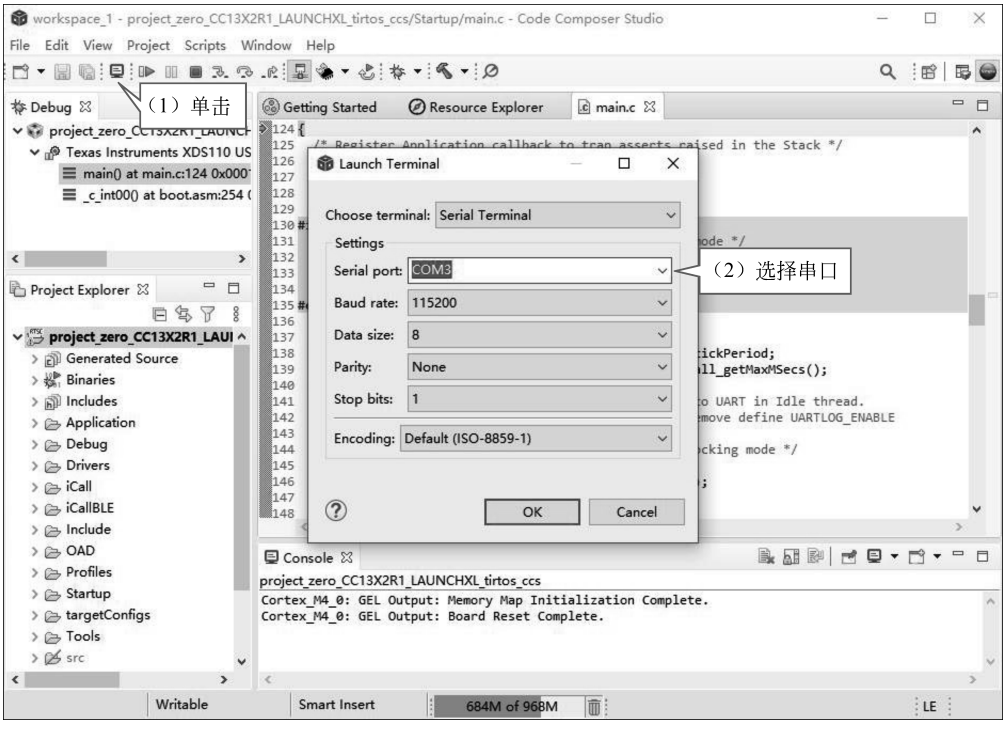


图 3.19　打开串口终端窗口