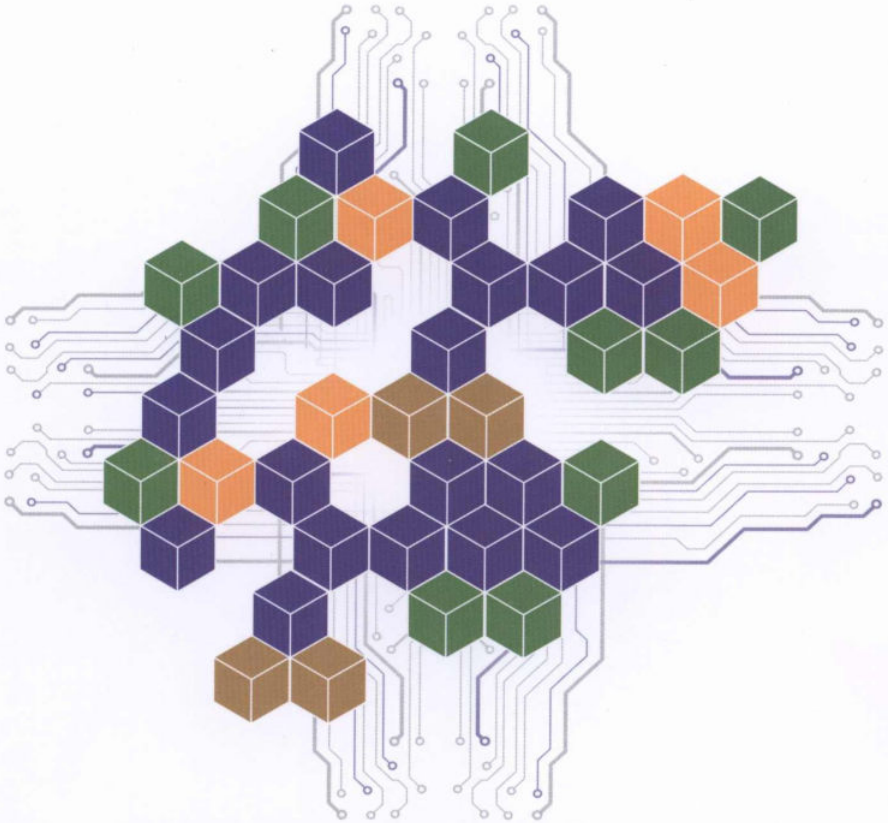
Broadview®
www.broadview.com.cn



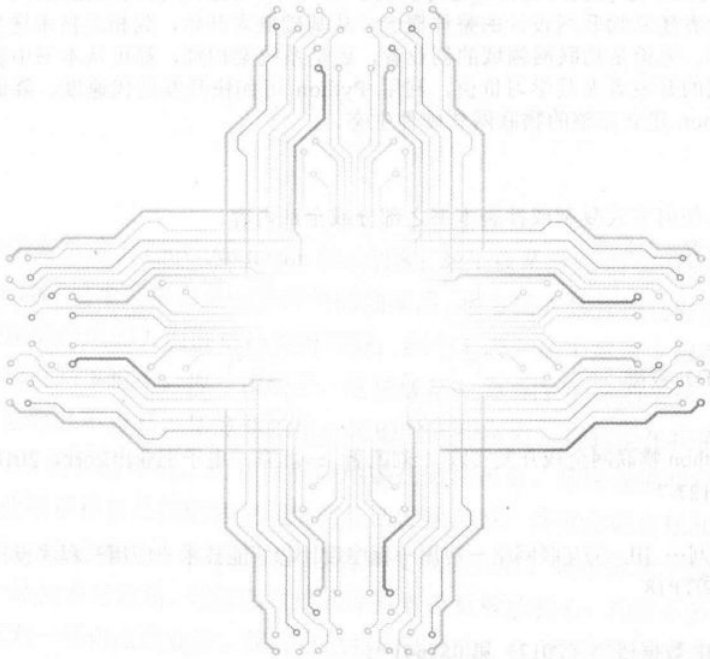
从芯片到云端
Python物联网全栈开发实践

刘凯 / 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

刘 凯

服务于微电子行业二十余载的资深工程师。曾在飞利浦从事软、硬件开发与产品设计等工作，有用汇编/C/C++开发嵌入式系统固件、用Perl/Python脚本做开发支持工具、用PHP/Java/Python做设备云和Web应用的丰富经验。现专业从事物联网相关项目设计和咨询服务工作。



从芯片到云端
Python物联网全栈开发实践

刘凯 / 著

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

内 容 简 介

物联网开发重新定义了“全栈开发”的范围。Python 作为一门快速发展的语言，已经成为系统集成领域的优选语言之一，其可覆盖从电路逻辑设计到大数据分析的物联网端到端开发。各领域开发者可以利用 Python 交叉涉足物联网设备、边缘计算、云计算、数据分析的工程设计。

本书尝试让读者建立物联网设计的整体概念，从基础概念开始，到相关技术选型、开源工程、参考设计与经验分享。无论是物联网领域的创业者，还是系统架构师，都可从本书中获得灵感。本书对于嵌入式开发领域的开发者尤具学习价值，利用 Python 可加快开发迭代速度、降低开发成本，并可以基于嵌入式 Python 建立完整的物联网软硬件生态。

图书在版编目（CIP）数据

从芯片到云端：Python 物联网全栈开发实践 / 刘凯著. —北京：电子工业出版社，2018.1
ISBN 978-7-121-31127-7

Ⅰ. ①从… Ⅱ. ①刘… Ⅲ. ①互联网络－应用－程序设计②智能技术－应用－程序设计
Ⅳ. ①TP393.409 ②TP18

中国版本图书馆 CIP 数据核字（2017）第 057641 号

策划编辑：张春雨
责任编辑：李云静
印　　刷：三河市双峰印刷装订有限公司
装　　订：三河市双峰印刷装订有限公司
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：45.25　字数：1012 千字
版　　次：2018 年 1 月第 1 版
印　　次：2018 年 1 月第 1 次印刷
定　　价：119.00 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

前几年国内引进了 Chris Anderson 的《创客：新工业革命》。打那时候开始，国内流行起“创客”风潮。“创客”这个词果真是一个洋气的舶来品，很多国人姑且把它看成硬件创业的预备役。但是大洋彼岸原产地的人们倒是朴实得可爱：织个毛衣，搞个室内大棚蔬菜。当然高科技类的自然少不了捣鼓一下机床，焊一块板子，这更像是一种 DIY 的怀旧文化：更加纯粹和快乐。做一名纯粹的创客并不容易，毕竟要抽出一定的时间和精力。直到现在我依然惦记着自己那台完成了一半的 3D 粉末打印机，而它就静静地躺在储物箱里。那时候的我已经开始为创业做前期准备，但商业项目和自己在创客空间玩的东西没啥关系，终究自娱自乐和商业有差别。

遇见 Allan 的时候，他也在努力从创客转变成创业者。我很惊诧于他虽然技术娴熟，也曾负责 NXP 产品技术与市场，却依然对技术保持着孩童般的初心，真的不多见啊。离开 NXP 后，Allan 决定成为一名自由职业者。靠着自己扎实的技术，从前端到后台，从硬件到软件，他一个人搞起了物联网的项目和产品。我们时不时在线上谈论可行的产品和市场策略。虽然我对硬件不熟悉，但是由于自己当时就职于 PTC，拥有些许物联网后台软件的认知，就这样我们相互参照着学习，并努力将其付诸实践。

2015 年 5 月，我离开了 PTC 并投身于机器视觉领域的创业，但依然保有对物联网的热情，尤其关注工业物联。而 Allan 在这几年的实践中积累了全栈开发的经验。终于有一天，他觉得是时候将他独自一人的全栈开发经验记录下来，并传播给这个领域的开发者了。我自然非常支持他，但是独自写一本技术类的书，这是多么考验人呀。之后和 Allan 的交流变少了，我想象得出他独自在房间码格子的情形。半年后，这本书的初稿终于扎扎实实地完成了。

创业者和分析师们总爱重复地问一个问题：物联网的风口有没有来？我们很难精准地去预判某个时间节点，但假如物联网是一个不远不近的方向的话，我们当下唯一能够做的便是顺着产业的脉搏而跳动。类比一下 PC 和移动互联网，我们依然处于物联网大规模商业化的早期。但是最终我们会迎来万物互联。让我激动的是万物互联的基础架构成熟后，在各个行业以及各个利基市场将会涌现出各种“新物种”，推动着商业和产业进一步提高效率，进一步打破边界。而对于希望投身于这个行业的技术人员来说，应该尽量抛弃这些华丽的时髦术语，回归技术本身。这本书平实地记录了读者需要了解和掌握的基础知识；与此同时，它从单一语言全栈开发

的概念出发梳理了一个完整的流程，而全局观的梳理能够更好地帮助技术人员去理解技术的本质。

技术总是在飞速地发展，书本记载的技能需要不断地升级更新。但是我能感受到 Allan 更希望传达的创客精神。创客愿意从零开始建一栋楼，他们或许不能建成一座摩天大厦，但至少也会筑成一幢别具一格的小楼房。这种纯粹的乐趣只有从动手实践中才体会得出来。但人们的生活节奏总是很匆忙，有这么一本类似于“宝典”的书，可以加快看官您动手的速度和效率。但愿您能享受从零开始搭建一个物联网项目或者产品的过程。

张成

浸梦信息科技有限公司创始人

自序

笔者曾经长期服务于微电子行业，现在从事物联网相关项目设计和咨询服务。

1995 年毕业后，笔者加入了飞利浦半导体上海技术中心，任软件工程师。在此期间的主要工作是使用汇编语言为国内客户进行各类显像管（CRT）彩电的固件开发。当时的技术环境，8051 都已经非常普及了，而飞利浦半导体彩电和固定电话技术方案中的控制器却依旧采用老旧的 Intel 8048 内核 MCU。该内核架构有许多限制：比如超过 2KB 代码需要切换代码段，缺乏高级语言支持，等等。虽然架构古老，但这个业务却一直是当时飞利浦半导体的“现金牛”（即主要利润来源）。

笔者后转入产品市场部，在那里可以接触到许多炙手可热的产品线。其中，笔者负责的产品如下。

- 8051 控制器：配合中国合作伙伴，如（北航）中国单片机实验室、南京万利、南京伟福、广州周立功等单位，合作推广 LPC764/9XX 及后来的 ARM LPC 系列。
- 通信产品：8048 内核电话机 MCU、传呼机、DECT 和中国数字无绳电话芯片组。
- 智能卡产品：包括电话卡、CPU 卡，以及最著名的 Mifare/Hitag RFID 和车用防盗钥匙。
- DSP 产品：Trimedia VLIW（超长指令集）DSP，用于视频电话和媒体处理。
- CPLD：低功耗 CoolRunner CPLD，后转售给 Xilinx。
- PDA：基于 MIPS R3000 内核的 Windows CE PDA 方案。

后来，笔者又重新拾起软件开发的工作，主要负责基于 8051/MIPS 的 LCD/DTV 的客户化固件开发。

应该这么说，在飞利浦的从业经验使得笔者积累了嵌入式开发经验，开阔了产品线视野，并积累了多方面的技术兴趣和行业人脉。同时，在开发这些嵌入式产品的过程中，笔者开始采用各类脚本语言来做代码生成和其他开发工具。

2008 年，飞利浦半导体部独立成为 NXP 公司之后，笔者开始了自己的创业之路。到目前为止，笔者独立设计过以下产品和参考设计：

- 基于 Cypress PSoC 的 RFID/UART/GPRS/TPMS 模块（C）。
- 基于 SDIO 闪存卡的 NFC 接口（FPGA/CPLD）。

- Wi-Fi 强制门户及热点分享网站（PHP）。
- GPRS+GPS AVL 设备及网站（C/Java/PHP）。
- TI C2800 DSP ANC 主动噪声抑制系统（C/ASM）。
- 网络爬虫，用于抓取超市的 POP 海报分发（Python+PHP）。
- Android 翻译 APP（Java）。
- 电子货架标签系统，第一个从设备到 APP 的完整原型设计（C+Python 网关）。
- GAP 创客电子模块，基于 NXP/Freescale/ST 的 M0/M3 处理器，并提供 Bootloader 和 ISP 软件（C/C++/Python）。
- 工业门物联网（C/C++/Python）。
- 呼吸机物联网（Python/Golang）。
- 电梯物联网（Python）。
- EPD 电子模块（C++/Python）。
- RFID 分类钱包（国家实用新型专利，已授权）。
- GPS 资产定位系统（C++/Python）。
- 电信 CDMA 基站监控设备（C++/Python）。
- VoLTE 高清语音监控设备（C++/Python）。
- 分级基金及股票监控报警系统（Python）。

离开 NXP 之后，笔者的设计不再受限于原公司的技术平台所涉及的消费电子产品领域，而是扩大到了互联网与物联网领域。笔者的个人体验是，无论是设备端还是服务器端，都有许多技术可以深入学习。但是两者融合，技术复杂度却呈现几何级数上升。

不同领域有不同的优势语言。一般来说，CPLD/FPGA 使用 VHDL/Verilog，MCU/SoC 固件开发使用 C/C++，桌面开发使用 C#/VB 等，服务器开发使用 Java/PHP/JavaScript/Python/Golang，手机 APP 使用 Java/Objective-C。

所以，笔者在工程实践中，一直在使用汇编语言/C/C++开发嵌入式系统固件，并使用 Perl/Python 脚本做开发支持工具，同时采用 PHP/Java/Python 做设备云和 Web 应用。一个完整的物联网应用涵盖许多环节：从数字逻辑电路设计，到硬件设计、固件设计、网关软件设计、服务器软件和网页设计、APP 设计，甚至模具的 3D 设计。出于工作的需要，即使环节长，笔者也不得不像“万金油”一样，亲自参与全过程的设计工作。虽然无奈，但笔者的修炼结果是，比一般硬件团队略懂服务器开发，比一般服务器/APP 开发团队略懂硬件开发，而且大致了解了物联网的许多具体技术。

笔者的个人体会是，物联网环节太长了！无论是设计、编码还是调试，物联网的庞杂特性都非常明显。首先设计和编码时间就很长，尤其在系统联合调试时，需要使用多种开发工具（仿真器、目标硬件、仪表、服务器、Web 控制台）。在这个阶段，有时候需要多台计算机才能够完

成调试任务。

以超市货架管理项目为例，其涉及 WSN 协议规划、节点端和网关端设备的固件开发和协议实现、服务器设计、手机 APP、条形码和二维码扫描。此项目笔者整整开发了一年才交付给客户，而且调试起来还挺麻烦。

个人单枪匹马，精力有限，无法同时兼顾所有环节，因而开发的项目格局不会太大。物联网开发应该为**团队合作**，甚至多个团队之间进行合作。每个团队对于各自的环节负责，做到接口标准化。这样才能够复用已有的经验和模式，并充分发挥其边际效应。即便是团队合作，也需要将自己使用的工具数量降低到最少，至少需要寻找到覆盖面较广的工具来开发。这也是现在许多“全栈”开发的目的。

采用单一语言做全栈开发

全栈开发最初出现在互联网行业，指的是能够同时开发网页前端和服务器后端。这包括能够做全栈开发的技术和掌握这些技术的工程师。该行业最典型的全栈开发语言是 JavaScript。

在物联网行业中，全栈开发的含义被延伸了。笔者推荐以 Python 作为全栈开发语言。本书的全栈开发涉及 IC（集成电路）设计、设备端（电路和系统）、服务器（含网页）端，以及移动端和数据分析端。使用单一语言可以多方面降低成本：

- 学习周期短，降低人力成本。
- 交付时间短，降低开发成本。
- 人力资源供应充分，降低人均开发成本。
- 容易形成生态，构建开发者生态圈，实现众包。
- 代码复用性强，代码可重用，开源市场有不少现成方案，可降低总体开发成本。
- 设备可以虚拟化，物理设备可以通过同一代码模拟出来，以加快工程启动周期，降低开发成本额和减少开发者间的责任推诿。

综上，物联网开发涉及面庞杂，开发周期长，寻找一种覆盖面广的编程语言和方法对企业和开发团队有现实意义。

Python 用于全栈开发

在笔者眼里，承担全栈开发的语言可以是 Java，也可以是 JavaScript，还可以是 Python。由于互联网的发展，加之 JavaScript 在前端语言中的优势地位，使得它开始延伸到了服务器后端和设备端。而 Java 原本就在设备端和服务器端都很有优势。从发展历史上看，在嵌入式平台中最早出现的是 Java，最近才开始出现 JavaScript 和 Lua 等动态语言。这都是服务器端企图深入到嵌入式行业的努力。至于 Python，由于其胶水特性，虽然性能不占优势，但是开发速度快，

比较适合做全栈的原型开发。

之所以出现企业和手机开发者力推 Java 开发，前端开发者力推 Node.js 开发前后端技术，某些群体力推 Go 的现象，除了技术本身的因素，许多情况下也是其教育背景和从业经历所导致的，即所谓出身和基因所决定的。在此，个人经历决定了笔者选择 Python 作为自己的主力开发语言。

曾经看过一个关于如何在 Java/JavaScript/C#/Python/Golang 等几种语言中选择一种作为主力编程语言的漫画式流程图。其中有一个选择：如果你喜欢乐高，那么请选择 Python。仔细想想，Python 的确很像乐高：

- 接口一致性高。
- 粗颗粒，构建速度快，适合原型。
- 有标准构件，如各种标准积木和标准库。
- 具备大量的定制构件。乐高中存在定制的主题人物和机器人组件，而 Python 也有大量的 C 扩展库和第三方应用库。

在我国的港台地区术语中，将 Integrated Circuit 翻译为“积体电路”，即积木化的电路。而新出现的各类集成技术，如 SoC/SiP，即系统芯片和系统封装，也是通过在电路 IP 领域和封装领域的创新来实现更大规模的电路整合。换而言之，不同规模的电路都是搭积木搭出来的。所以，半导体行业应该会比较偏爱类似于积木的 Python 语言。

回顾自己的从业和工程经历，大概以下是笔者偏爱 Python 开发的原因：

- 在固件开发中，接触到使用脚本语言（gawk）来设计代码生成器简化开发。
- 电子工程经验，接触并了解了许多企业的设备联网需求。
- 互联网工程开发经验，接触到了互联网/物联网领域的诸多环节。
- 在网站和 APP 开发经验中，不得不使用多种编程语言用于软件开发，了解工程管理的痛点。

Python 作为一种胶水语言，可在物联网及嵌入式系统中承担大量任务，并可以部分替代 VHDL/C/C++/Java/PHP/JavaScript 等各类语言，或者与这些语言进行互相调用。但是让一位工程师抛弃原有技术栈换用其他语言是困难的。最初，笔者只是在工程实践中发现 Python 的“出镜率”相当高。在一些小场景中笔者尝试使用 Python 开发后，积累了一定的使用经验。后来为了加速开发，笔者开始在客户工程中大量使用 Python 进行原型验证和服务器端开发。最终，Python 成为笔者的主力开发语言。

说起来，笔者本人的经历与 Python 的胶水特性很类似。笔者不能算是 Python 高手，所有的开发都是仅仅读了最基本的演示代码后就立即着手进行工程开发。笔者甚至连相关基本入门书都没有看完整，就着手使用 Python 构建系统。缺乏耐心的代价就是不断重新造轮子，即所谓的“重构”。不过，在不断换“轮子”的过程中笔者充分体会到了 Python 的各种优点。

无论是面向过程（POP）、面向对象（OOP）、面向切面（AOP），还是更加抽象的函数式编程（FP），Python 都可以支持。编程思想只有在项目中才能被不断加深理解。这一点，Python 对笔者的帮助非常大。之前，虽然也写过 C++/Java 程序，但是实际上真正让笔者完成从面向过程到面向对象编程思维转换的语言恰恰是 Python。

相当多的 Python 代码，一开始编写的时候是采用各类函数的面向控制编程，脚本化的倾向很强。随着代码复杂度的增加，笔者不得不反复重构代码，并主动引入了 OOP 的编程方法。体会了 OOP 的好处后，促使笔者反过来在设备端设计中重构 C++代码。在物联网服务器端开发时，笔者接受了面向切面的概念。在编写本书的时候，笔者又学习了函数式编程。在以后的开发中，笔者会有意识地增加更加抽象的编程思想以简化日常的编程设计。

实际上，在从事物联网的后端设计时，许多朋友强力推荐笔者使用 Java 进行开发。因为 Java 在企业级应用中积累了许多可重用的设计，是企业级应用的首选语言。但是笔者接触并熟悉 Python 之后，坚持使用 Python 开发网关和服务器。笔者发现使用什么语言真的不那么重要，只要自己熟悉就好。况且 Python 还可以用 Jython 来对接 Java 重用 Java 资源。说起来 Python 和 Java 是两种极端：Python 可以在许多语言中实现，而许多语言利用 Java VM 来运行。

熟悉了 Python 后，笔者就在日常工程中坚持使用 Python：在端口扩展与仿真、代码和文档生成、Web/IoT 服务器及嵌入式平台 Python VM 中都可以用到。而且每次开发后，总能够保留一些 Python 工具提交给开源社群，或者以后自己用。这也是不断自我强化的过程：熟悉一种工具，就会不断地利用这种工具去解决问题。

Python 的缺点及应对措施

首先，许多开发者认为 Python 的运行速度较慢，尤其无法与 C/C++编译的原生代码相比。由于 VM 的设计架构不同，Python，尤其是 CPython 比 Java/Lua 还要慢。Python 作为一种开源的语言，即使有各种各样的问题，利用开源社群的力量也可以更容易地找到各种解决方案。现在使用 JIT 技术的 PyPy 加速已经非常成熟，在许多场合都可以应用。Cython 也是一种性能极高的扩展，可以实现与 Golang 类似的性能。配合 libuv 异步库，Python 的网络性能不输于任何一种编程语言。性能不是唯一的要素，Python 的强大在于：**生态的完整，开发速度快，运行速度也很快。**

其次，Python 语言和代码本质上是开源的，所以更加适合开源软件使用。如果要实现闭源的商业化软件，可以将 Python 源码编译成 pyc，或使用各类 C 和其他扩展帮助保护核心设计，其代价是损失了 Python 跨平台的特性（除非扩展中也采用了某种跨平台技术，比如 JVM）。

最后，Python 的 GIL 问题也很有名，对多线程设计不利。解决方法有很多：多进程、协程及其他 Python 实现（如 Jython、PyPy、Cython 等）均可以回避这个问题。

前言

本书讲述如何以 Python 为主要编程语言，实现“从芯片到云端”的物联网应用系统快速开发和系统扩展。通过阅读本书，读者可以充分体会 Python 作为一门全栈开发语言，是如何在物联网的设备端、应用端、服务器端和数据端环节中发挥作用的。

编写本书的初衷是为了让准备或者已经从事物联网开发的读者能够通过 Python 语言缩短相关学习和开发周期；同时与大家分享一些经验教训，希望能够让读者在具体开发中回避各种“坑”。这不仅对开发团队，对于企业甚至投资者决策也是有益的。

大多数物联网相关书籍比较关注物联网系统和服务器端设计，但是物联网与互联网的设计差别在于：物联网系统设计受限于有限的设备计算能力、巨大的连接数量、独特的数据特征。所以完整的物联网系统设计需要考虑的要素比互联网更多，需要掌握的知识面既广且深。如何在短期内实现系统上线，并安全、平滑地实现规模扩展，一直是大家思考的问题。开发者可以采用的对策如下：

- 减少开发语言和工具种类。
- 使用成熟的参考设计和编程框架。
- 使用主流的云计算服务和可扩展的系统设计。
- 开源硬件、软件设计和并行开发模式。

有许多事情“开弓没有回头箭”。物联网的最大特点是大量的定制需求，而且上下环节的衔接往往存在技术依赖性，某个环节的决定往往会对其他环节的实施带来很大的影响，并可能造成开发团队间的责任推诿。这需要系统设计者事前做许多调研功课。笔者专注于设备域和服务器域，但本书力求带来更宽的视野，包括物联网相关的应用、产品和生态，介绍不同的系统架构和云计算服务，并在不同的技术选项中推荐几种比较适合工程实施和实际需求的主流组合。

在收集资料的过程中，笔者发现 Python 作为一门通用编程语言，应用范围非常宽泛。相信本书内容中有许多物联网相关的 Python 应用是出乎大多数人意料的：

- 支持 SPICE/IBIS 仿真与 VHDL 设计和电路的自动测试。
- 可以在许多流行的 8/16/32 MCU 上运行，包括 AVR/PIC/ARM/MIPS。
- 支持绝大多数 MCU/MPU/CPU 的外设和工业总线，而且编程接口非常灵活。

- 可以在各种类型的 Linux 上运行多种 Python 运行环境，包括 CPython、Jython 和各类嵌入式 Python。
- 通过 Jython 运行于 Java Runtime 中，与 Java 类库完美结合，切入企业级应用和大数据分析。
- 可以跨平台开发桌面应用和手机应用。
- 大量现成的网络安全和分析工具，可帮助开发者定位通信报文错误，或寻找系统安全漏洞。
- 提供大量的辅助工具，包括文档、软件工程、虚拟仪器、媒体处理等，为此笔者特地预留了第 8 章进行罗列。
- 物联网网关、服务器架构、数据分析和可视化、虚拟设备、通信协议定制等领域开发效率超高。

从 SPICE/VHDL 开始，到服务器，Python 实现了“从芯片到云端”的全栈开发。笔者希望这些内容和案例能够帮助开发者在启动项目前对开发有全局性的了解，并做出正确选择。

同时，本书的写作过程采用了 Python 相关工具，也是“全栈开发”实例之一。

- 格式：将 Python 文档中常见的 reST/Markdown 作为基础书写格式。
- 编译：采用 Sphinx 将 reST 章节编译成流行的 HTML 网页、ePub 电子书籍。
- 转换：采用 Pandoc（Haskell）转换成交付给出版社的 docx 主流文档格式。

读者可以将本书看作单一编程语言的物联网应用小百科，通过书中的简单例子大致了解物联网的开发流程，并可以根据自己的兴趣，在每章的延伸阅读[1]清单中深入探索、掌握物联网开发技术的具体实现细节。

目标读者群

本书的目标读者群是以下两大类开发者。

- 互联网开发团队：熟悉移动端 APP 的开发、服务器架构和网页前端开发，但对于传统制造业的技术领域，如芯片设计、硬件设计、固件设计、硬件系统集成，以及批量生产和库存管理缺乏足够的了解。
- 设备开发团队：主要是传统制造业产业链中的半导体供应商、独立设计公司、设备制造商。他们熟悉硬件设备的设计和流程，但普遍对于互联网应用和物联网架构缺乏足够的了解。

当前的制造业变化趋势是，设计与平台标准化，导致产品同质化竞争严重。这使得传统制造业在市场中逐渐丧失了议价权和话语权，处于被整合的被动地位。这些企业和团队在物联网

1 因篇幅所限，各章“延伸阅读”部分放于 www.broadview.com.cn/31127，请自行下载。

时代异常焦虑，急需掌握数据接入和数据分析技术，以增加市场份额，并提升市场竞争力和议价权。本书第 9 章主要讲述物联网服务器后端开发，可以帮助传统制造业了解服务器端和数据端的发展趋势、大致的技术方案构成，并可以利用 Python 做些简单的设备测试。

除了工业物联网、行业物联网外，消费端智能硬件领域的物联网开发案例非常多，这是市场热点之一。许多创业团队虽然可以自行设计 APP，搭建服务器，但是团队往往缺乏设备端制造经验，并仍在各类硬件问题中艰苦跋涉，苦苦摸索。本书在第 4 章中介绍了成熟的元器件、连接模块和实时操作系统，配合 Python 快速原型开发能力，让项目可以快速上市之余，还可以为设备添加各种“智能”应用。

此外，许多读者可能希望从全局角度了解物联网应用、各类技术方案甄选标准，以及具体技术细节。本书也尽可能地进行罗列，并就一些常见问题特别加以说明。

总的来说，本书适合对物联网及相关热点，如智能硬件、工业 4.0、万物互联的应用与实现技术感兴趣的人群阅读。目标读者群除了互联网从业者、微电子和 OEM/ODM 制造商、应用系统集成商，还包括学生、教师、创客、极客、Python 语言爱好者、产品经理、项目经理，企业高管和创投基金经理等。

最低阅读要求

由于本书是一本技术书籍，因此需要读者具备一定的编程经验和技术热情。如果读者对于 Python 基本语法有一定的了解那就更棒了。即便没有 Python 的使用经验，相信 Python 易学易用的特点也可以让读者很快入门。

此外，由于代码中大部分采用英语注释，因此需要读者具备基本的英语阅读能力。

本书的目的

核心目的

- 为应用开发团队提供设备端硬件、固件开发流程和开发工具方面的工程建议，并提供一些可以用于与服务器对接的硬件平台和参考设计。
- 为设备开发团队，提供服务器前后端/移动端的系统架构、开发框架、生态平台方面的工程开发建议，提供可以不断升级的可扩展架构和开发路径，以满足产品从原型测试、中试、量产到分布式规模生产系统整个产品生命周期的需求。

其他目的

- 分享基于 IaaS/PaaS 云计算平台的服务器开发经验，包括设备云、应用云和大数据服务。
- 分享可快速部署的物联网网关（Gateway）、边缘服务器（Edge Server）原型设计。

- 汇集 Python 在计算机系统中方方面面的应用信息，并持续更新。
- 吸引各方合力推动 Python 在嵌入式虚拟机/网关/服务器/大数据分析方面的开源活动。

本书内容安排

物联网环节长、技术庞杂，涉及的每种技术领域都值得大家仔细钻研学习。可以这么说，许多话题和技术都可以单独出一本书。所以本书力求在有限的篇幅内，突出物联网特征并使用 Python 落地生根，使之成为快速开发迭代的基础；与此同时提示在 Python 应用中可能遇到的问题和解决方案，以降低读者的学习成本。

本书以数据的流动方向，即数据的设备端采集、服务器接入、转发、分析到用户端的呈现为主线，并以 Python 语言从入门到各个技术栈中的应用作为辅线来安排章节。

本书内容编排经过多次斟酌和修改，最终按照以下顺序介绍。

章　节	简　介
第 1 章　物联网简介	概述物联网的定义、发展趋势以及物联网应用与技术等
第 2 章　Python 语言基础	数据类型、数据结构、内置函数和标准库
第 3 章　Python 语言进阶	多种实现、与其他语言的接口、物联网常见技巧
第 4 章　嵌入式系统开发	数字逻辑与模拟电路设计、C/C++固件开发以及主流的平台与供应商
第 5 章　设备连接和编程接口	物联网的多种连接性与编程接口以及 Python 支持包
第 6 章　嵌入式 Python 虚拟机	深嵌入式、嵌入式 Linux 最小系统以及各类 Python 虚拟机实现、演示
第 7 章　Python 应用 APP	在主流桌面操作系统和移动端中的 Python APP 开发
第 8 章　Python 开发辅助支持	在物联网开发环节中的原型验证、虚拟设备、数据分析等多个方面的 Python 开发工具
第 9 章　物联网服务器端设计	物联网网关、边缘服务器、Web/IoT 服务、开发框架和连接选项
第 10 章　融合应用与数据分析	科学计算、数据统计、数据挖掘和大数据分析平台和工具，以及数据可视化

除了本书内容，笔者还整理了诸多书中提到的 Python 扩展包和演示代码，并计划依托出版社网站和其他互联网服务进行分发。本书为笔者一个人写就，缺少专家进行校对，本人水平有限，书中难免有疏漏、错误，欢迎读者指正。但笔者精力亦有限，无法一一回复，祈谅。

本书未包括的内容

因为篇幅的限制，也因为物联网的特性，所以本书安排的内容比较繁杂。本书未能针对特定硬件、软件、云服务展开，也没有针对物联网提供完整的开源设计。这些希望读者在书本之外展开。本书出版后会依托各类互联网服务（如 GitHub、社交网站和 BBS）展开后续的开源设计活动。

软硬件环境

除非特殊应用和声明，本书主要的操作环境为 Windows 7（64 位）及 Ubuntu Linux 12.04（32/64 位）。Python 版本为 V2.7.11 和 V3.5。

在微控制器方面笔者推荐 ARM mbed 兼容的 LPC/STM/KL 开发板，或 Arduino；对于卡片电脑，推荐树莓派或者兼容的国产 Linux SBC；对于 MicroPython，推荐在 STM32F4XX/ESP8266 开发板上运行。

版权声明

感谢

本书付梓需要感谢许多机构和个人。

- 知乎网站：本书的创作主题来源于笔者在知乎上的提问，并得到了知乎网友（包括出版社编辑）的热心解答、正面鼓励和推动才能够走到这一步。
- 张春雨先生（永恒的侠少），电子工业出版社的策划编辑：在知乎上遇见后，你一路推动本书的出版。感谢你的耐心和鼓励。
- 张成先生，物联网创业伙伴：你不断鼓励笔者继续深入物联网开发，并拨冗为本书作序。
- 李云静编辑在本书成书过程中对我这一新手作者给予了极大的耐心，反复校对、纠错，感谢你和团队的付出。
- EEWorld 编辑 nmg 和版主 dcexpert：你们提供了宝贵的 MicroPython pyboard 开发板。
- 诸多开源项目的作者们：感谢大家对于开源软硬件项目的热情和不厌其烦解答问题的耐心。

芯片供应商及分销商：

- NXP（恩智浦）——感谢免费提供 LPC 系列开发板。
- Freescale（飞思卡尔，已与 NXP 合并）——感谢慷慨提供大量 KL25 的样片及技术支持。
- TI（德州仪器）及分销商 Serial（新晔科技）——感谢提供 WSN 技术支持。
- Cypress——感谢免费提供 PSoC 开发板。
- Fujitsu（富士通）——感谢友情提供 FeRAM RFID。