

预览版

START

微软经典 IoT 课程中文版

深入浅出 IoT: 完整项目通关实战

IoT for Beginners

[美] 珍·福克斯 (Jen Fox) [美] 珍·卢珀 (Jen Looper) 著
[英] 吉姆·贝内特 (Jim Bennett)

柴火创客空间 译



清华大学出版社



物联网简介

- 什么是“物联网”
- 物联网设备
- 准备好你的设备
- 物联网的应用
- 探究你身边的物联网设备

1



深入了解!

- 物联网应用的组成部分
- 深入了解微控制器
- 深入了解单板计算机

2



迁移



检测

土壤水分与
传感器校准

- 土壤水分
- 传感器如何与物
联网设备通信
- 测量数据
- 校准传感器

6



自动化

浇灌
植物

- 用低功耗物联网设备
控制高功率设备
- 控制一个继电器
- 通过 MQTT 控制你
的植物
- 传感器和执行器的
计时
- 添加对浇水周期时
间的控制

7



位置追踪

- 联网车辆
- 地理空间坐标
- 全球定位系统 (GPS)
- 读取 GPS 传感器数据
- NMEA 协议 GPS 数据
- GPS 传感器
数据解码

11



存储位置数据

- 结构化和非结构化的数据
- 将 GPS 数据发送到 IoT 中心
- 热、温、冷数据路径
- 使用 Serverless 代码
处理 GPS 事件
- Azure 存储账户
- 将 Serverless
代码与存储相连

12



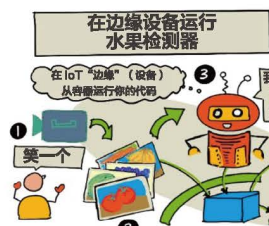
可视化位置数据

用 IoT 设备检查水果质量



- 相机传感器
- 使用 IoT 设备捕捉
图像
- 发布你的图像分类
器
- 从 IoT 设备对图像
进行分类
- 改进模型

16



在边缘设备运行 水果检测器

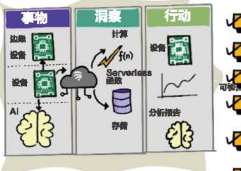
- 在 IoT “边缘” (设备)
从特定制行的代码
- 我预测这个
水果熟了!
- 水果已
成熟!
- GET MATH

17

- 边缘计算
- Azure IoT Edge
- 注册 IoT 边缘
设备
- 设置 IoT 边缘
设备
- 在边缘设备运
行你的模型

17

从传感器触发水果



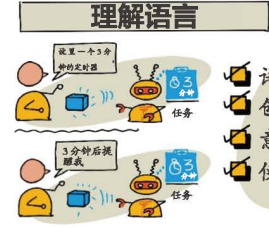
用 IoT 设备识别语音



- 麦克风
- 从 IoT 设备
捕获音频
- 从语音到文本
- 将语音转换为文本

21

模拟音频经采样转换为数字音频



理解语言

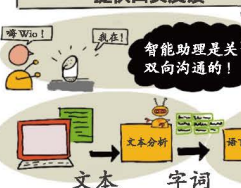
- 设置一个 3 分
钟的定时器
- 任务
- 3 分钟后提
醒我
- 任务

22

- 语言理解
- 创建一个语言理解模型
- 意向和实体
- 使用语言理解模型

22

设置定时器并 提供口头反馈



交互

质世界交互

- 传感器 收集数据
- 执行器 发送反馈
- 项目 制作一个小夜灯

3

连接

将你的设备连到互联网

学会

- 发送与接收消息
- 连接小夜灯到 MQTT 代理
- 连接设备到互联网

4

预测

植物生长和温度数据有关

用温度传感器 预测植物生长

5

种植数据上云

- 什么是云
- 创建一个云订阅
- 云物联网服务
- 在云中创建一个物联网服务
- 与 IoT 中心通信
- 将设备连接到物联网服务

8

迁移

应用逻辑上云

- 什么是 Serverless
- 创建一个 Serverless 应用程序
- 创建一个 IoT 中心事件触发器
- 发送方法请求
- 将你的 Serverless 代码部署到云

9

确保你的植物安全

- 为什么你需要保护物联网设备
- 密码学
- 保护你的物联网设备
- 生成和使用 X.509 证书

10

地理围栏

- 什么是数据可视化
- 地图服务
- 建一个 Azure 地图资源
- 在网页上显示地图
- GeoJSON 格式
- 用 GeoJSON 绘制 GPS 数据

13

地理围栏

- 什么是地理围栏
- 定义一个地理围栏
- 针对地理围栏的测试点
- 从 Serverless 代码中使用地理围栏

14

训练水果质量检测器

- 使用 AI 和 ML 对食物进行分类
- 通过机器学习进行图像分类
- 训练一个图像分类器
- 测试你的图像分类器
- 重新训练你的图像分类器

15

质量检测

从传感器

- 构建复杂 IoT 应用
- 设计一个水果质量控制系统
- 用传感器触发水果质量检测
- 将数据用于水果质量检测器
- 使用开发者设备模拟多个物联网设备
- 应用于生产

18

训练一个库存检测器

盘点店内库存

零售

训练和使用对象检测

另一个自定义视觉场景

- 对象检测
- 在零售中用对象检测
- 训练一个对象检测器
- 测试对象检测器
- 重新训练你的对象检测器

19

从物联网设备上检查库存

过道上架要补货 10 个

大促销

- 库存盘点
- 从你的 IoT 设备中调用对象检测器
- 边界框
- 重新训练模型
- 计算库存

20

支持多语言

- 文本转语音技术介绍
- 设置定时器
- 将文本转换为语音

23

支持多语言

- 翻译文本
- 创建翻译器资源
- 翻译服务
- 应用程序中用翻译支持多语言

24

恭喜结业

You made it!!
#IOT4Beginners

CREATED BY
@SKETCH THE DOCS

内 容 简 介

本书以通俗易懂的方式,全面讲述物联网基础知识。全书贯穿一个大项目,涉及种植、运输、制造、零售和居家等常用的物联网场景,涵盖 IoT 的核心概念、设备和传感器的使用、数据的收集和分析、边缘计算,以及如何通过微软 Azure 云服务实现远程控制、通过语音与设备交互等内容。每一课都包括理论知识和实践项目,帮助读者深入理解和掌握物联网的基础知识。

本书提供详细的代码示例和操作指南,让读者能够轻松上手实践。书中项目同时提供基于 Arduino 和树莓派两个不同平台的硬件套件实现示例,还包括使用虚拟硬件学习的示例,以满足读者的不同需求。

本书适合物联网相关专业的学生阅读,也适合对物联网有兴趣的自学者阅读。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。举报:010-62782989, beiqinquan@tup.tsinghua.edu.cn。

图书在版编目(CIP)数据

深入浅出 IoT:完整项目通关实战/(美)珍·福克斯(Jen Fox),(美)珍·卢珀(Jen Looper),(英)吉姆·贝内特(Jim Bennett)著;柴火创客空间译. —北京:清华大学出版社,2023.6

ISBN 978-7-302-63402-7

I. ①深… II. ①珍… ②珍… ③吉… ④柴… III. ①物联网 IV. ①TP393.4 ②TP18

中国国家版本馆 CIP 数据核字(2023)第 068600 号

责任编辑:王中英

封面设计:孟依卉

责任校对:徐俊伟

责任印制:丛怀宇

出版发行:清华大学出版社

网 址: <http://www.tup.com.cn>, <http://www.wqbook.com>

地 址:北京清华大学学研大厦 A 座

邮 编:100084

社 总 机:010-83470000

邮 购:010-62786544

投稿与读者服务:010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈:010-62772015, zhiliang@tup.tsinghua.edu.cn

印 装 者:北京嘉实印刷有限公司

经 销:全国新华书店

开 本:185mm×260mm

印 张:27.5

插 页:1

字 数:914 千字

版 次:2023 年 7 月第 1 版

印 次:2023 年 7 月第 1 次印刷

定 价:109.00 元

产品编号:098875-01

作者序

1874 年 11 月 30 日，巴黎的滑雪爱好者们被一项新的发明惊呆了。工程师们在勃朗峰上部署了传感器，通过一根电线穿越 300 英里连接到巴黎，他们突然可以获得勃朗峰的实时天气和雪深数据。这是遥测技术（telemetry，源自希腊语，意思是远距离测量）的最早例子之一。

让我们快进一个半世纪，如今遥测技术已无处不在，从机器上的传感器，到健身追踪器、智慧城市等。我们已经从单纯的收集数据，发展到建立一整套互联设备，利用云和人工智能的力量，从收集的数据中做出决策，优化生产线，节约能源，改善我们的健康。这些设备构成了今天的物联网。

物联网（IoT）是目前发展最快的技术领域之一。现在想要不被物联网设备包围，似乎越来越难了。当我坐在办公桌前写这篇文章时，我的房间连接了传感器来监测温度和空气质量，不断地将这些数据传送到控制供暖和制冷系统或管理空气净化器的算法中；通过语音助手我可以控制房间的灯光或打开车库门；家门口的智能门锁可以让我从世界任何地方看到谁在我家外面。

作为一名工程师，我认为物联网是一项非常有趣的技术。“物”是你可以触摸到的有形物品，在我看来，这增加了构建物联网项目的实用度。例如，你可以看到一个由土壤湿度数据控制的自动浇水系统，以便让作物长得更好。这种“物理计算”也使物联网成为孩子们学习技术和编程的好方法。

有大量的物联网设备和套件可供选择，从业余爱好者的一体式开发套件到为工业用途设计的专业板和传感器。你可以在互联网上找到构建不同解决方案或使用一系列设备和传感器的教程。但是，一直缺少的是真正高质量的文档，它可以教你所有关于物联网的知识，包括基本原理、用例、硬件和软件。这是我想填补的一个空白。

当我开始着手在微软创建 IoT for Beginners 课程时，我在内心给自己设了几个目标。首先，我想在课程里提供物联网的基础知识。有许多针对特定物联网设备或云服务的优秀教程，但我找不到任何涵盖所有基础知识的教程。其次，我想给学习者提供多种硬件选择，包括使用虚拟硬件学习，以避免潜在的高成本。我非常感谢矽递科技（Seeed Studio），它帮我提供了易于购买的工具包和所需的硬件。第三，我希望这本书的水平适用于高等教育水平的学生，或者高中的优秀学生。为此我很感谢一群来自 Microsoft Learn 的学生大使，他们帮助我审查了内容，以确保内容处于正确的难度水平。

这本书包含了你在激动人心的物联网世界中开始探索所需的一切。我希望这能帮助你学习所有你需要知道的知识，并激励你创造令人赞叹的项目。

Jim Bennett
IoT for Beginners 的主要作者之一

推荐序一

为什么需要学习物联网

物联网不是一个新的概念。以人们对于技术的追赶和通常的学习逻辑来看，新技术似乎永远是最时髦的，我们应该忘掉物联网，去努力追赶当前最新颖的热门技术和话题，诸如 AIGC（Artificial Intelligence Generated Content，基于人工智能生成内容），元宇宙、Web3.0、NFT 等，似乎只有这样才能让我们赶上时代的潮流。

但是，新的就一定是好的吗？时髦的就一定是有用的吗？或者说，如果我们学习技术的目的是为人类社会产生更大的福祉，那么除了追赶所谓的技术时尚以外，什么样的知识能够让我们真正提升自身的技术水准，最大程度地拥有造福社会的能力？这是一个更为基本的问题。

如果我们认真地梳理当下这个数智时代的技术发展脉络，大概会发现人类即将进入一个全新的人机协同与共生的时代。近几百年来，人类先是通过借助机器的机械力来提升个人乃至人类社会整体的生存与发展空间，现在不仅需要依赖机器的机械力，还需要通过学习和掌握机器的计算能力来提升自身的生存与发展潜力。

在以工业革命为代表的机器时代早期，那些学会操作机器和驾驶汽车的人获得了生存与发展的优势与先机，这种机器能力可称为机械力，人类社会因为积极拥抱了这种机械力而获得了指数级别的增长与发展。之后人类又经历了电子与信息时代。从近来愈发成熟的“thinking machine”（思考机器，借用基辛格在《大西洋月刊》发表文章中的定义）的发展来看，这些具备计算能力的机器通过高效处理数据而慢慢展现出某种类似于人类智能的能力。这种被称之为“人工智能”或者“机器学习”的新型机器能力，虽然如同机械力机器一样，在早期尚不够完善，但越来越显露出这种思考机器可以强化人类能力的潜质。我们现在应该可以做出一个大概率会实现的推断，那就是当下那些主动拥抱并掌握这种新型的机器智能能力的人或组织，相比于那些拒绝或者放弃学习并掌握这种能力的人或组织，会获得更大的生存与发展机会。这也就是为什么现在越来越多的人开始关注最新的技术，希望能够在在这个剧变的时代，具备最大化的生存与发展可能性。

那么问题就来了，每个人的时间与精力都是有限的，而当下技术的迅速发展并因此而产生的海量知识，已经远远超越了绝大多数人能够接受并理解的范畴，我们该如何在有限的时间与精力约束下，尽量学习到对自身的发展和社会的进步有用的知识，而且这种知识还要具备尽量长的有效期呢？

为了回答这个问题，首先我们要认识到，有时候人们为了体现出技术的进步，会经常发明一些新的名词来彰显现在与过去的不同。这种急于表达与过去不同的思维逻辑，让我们姑且称之为“求异存同”，如果应用在技术的创新方面有它的用武之地。当我们的关注点放在技术的落地实践和技术对人类社会产生应用价值时，“求同存异”则是一种更为实用的技术应用范式。这种思维方法不是急于探索哪些技术改变了，而是先搞清楚哪些技术“没有改变”，以及技术如何能够尽快地为人类所用，并产生出最大化的社会价值。

当然与此同时，我们也要不断地突破我们对技术理解的局限性，以创新的方式探索出全新的技术领域。

“求同存异”与“求异存同”这两种方式并不矛盾，一般来说，技术的发展都有它的客观规律，相同的部分大多是技术的根脉，不同的部分通常是在根脉之上长出的繁茂枝叶和花朵。它们是在不同的前提条件下的一体两面，是一种辩证统一的关系，只不过会影响到我们如何在海量的信息领域中学习到对我们有用的知识。通常而言，根脉型的知识少而精，不易掌握但也不会轻易改变；枝叶或花朵型的知识多而杂，容易掌握也容易变化。这是一个没有标准答案的问题，需要每个人认真思考并做出自己的选择。

以我个人的思维习惯，我通常都会采用被称为“五问法”或者说“苏格拉底方法”的追问方法，不断从“为什么”这个本源问题出发，尽量求本溯源地找到隐藏于当下五花八门的技术背后的根源，同时理解由这个根

源而长出的各种支脉之间的关系，慢慢构建起自己在这个即将到来的数智时代的知识体系，以最大程度地应对时代的挑战。

基于这种方法，我们会逐步理解到，机器的这种学习与智能能力，源于它通过计算能力对于数据的加工。而这种计算结果的作用则体现为，机器对于人类社会各种现象的分析与模式的识别。那么为了能够理解并掌握这种机器能力，我们需要知道数据是如何产生的，以及数据是以何种方式来表达人类社会的各种现象的；我们还需要知道这些数据是如何被机器计算的，以及这些计算结果是如何被应用的。人类社会数据的产生、传输、存储、计算和应用就是物联网的范畴，这也就是为什么我认为物联网的技术是这个数字时代的本源技术。物联网技术的覆盖面极其广泛，最重要的是，它解决了物理世界与数字世界之间的关联与协作问题，也是人类与机器之间的关联与协作问题。

根据不同的定义方式，它所覆盖的技术内涵与外延会略有不同，但是物联网技术总是与数字孪生、元宇宙、人工智能、传感、通信、存储和执行技术互为表里。因此，从物联网技术入手，不断地深入展开，更容易建立起一个完整的、适应数智时代特征的技术知识体系，有助于明白人类自身在这个新时代的定位和应该采取的行动。另外由于物联网技术跨越了人类所熟悉的物理世界和不是很熟悉的抽象数字世界，在学习的过程中，人们能同时学到如何将现实的物理世界映射进抽象的数字世界；同时又能够亲身体验到抽象的数字世界所计算的结果是如何影响物理世界的。这种真实的学习体验本身就是一种非常有趣的经历。

由物联网技术的特色而带来的有趣学习经历，在本书中体现得更加淋漓尽致。我们还是要明白，这本书里面所介绍的知识不是容易掌握的知识。但是通过作者清晰的描述和明确的实验步骤，读者很容易就能看到自己在数字世界里的努力，马上就体现为物理世界里的成果。这得益于物联网技术本身的特色，同时也得益于本书几位作者和国内译者丰富的技术传播经验。但是本书中实验所涵盖的技术范围博大精深，几乎囊括了所有与数据和人工智能相关的技术领域。就像大家在学校中常用的经典教材一样，要想把这本书传达的知识体系建立完备，读者需要根据书中的提示和相应的网站内容，进行大量扩展学习。

幸运的是，本书的知识讲解始终坚持理论与实验相结合，它采用技术界常用项目制方式进行组织，可谓学习体验生动活泼，学习效果立竿见影。以我自身的学习经验来判断，我相信只要读者们能够静下心来，一步一步地按照书中描述的实验步骤做下去，同时不要忽略其中任何一段扩展知识学习的机会，就能够建立起一整套完备的数智时代知识体系。作为一个物联网技术的拳拳服膺者，同时也是一个物联网技术的爱好者和实践者，我也预祝读者们能够借助本书，更加充满信心地迎接这个剧变时代的机遇与挑战。

韦青
微软（中国）首席技术官

推荐序二

世上本没有 IoT。

随着移动设备占领了人们的空间和时间，开发者自然将目光投到更广阔的天地和万物。从工业 4.0 到元宇宙，层出不穷的信息技术热门概念都试图拉近数字世界和物理世界的距离，更多的接入意味着更广阔的市场和更长的跑道，物联网理所当然成为各种新应用的共通的技术底座。

我们为新技术的涌现感到骄傲，但物联网并不是一个横空出世的石猴——绝大部分的技术都已经历过几十年甚至上百年的发展，曾经被归纳在仪器仪表、电子、自动化、通信工程等专业领域。物联网像一口大锅，里面翻滚着最新的传感器、通信、单板计算机、机器学习、新材料等软硬件技术栈，它们不断演化和融合，等待着用户的选择，以解决实际问题。

然而，来自各个传统行业的用户可能很难理解这一大堆晦涩难懂的技术，而且技术公司为了凸显自己的差异化，习惯于发明新奇的名词，更进一步阻碍了物联网技术的应用。过去十多年的创客运动和开源硬件潮流，让这个尴尬的局面得以改善。大众也开始扮演技术创造者的角色，让物联网技术走出实验室，变得随手可得，像玩具一样被大众肆意组合并展示。其中一部分创客们不再满足于有趣的发明，他们进入（回到）传统行业，将越来越成熟的技术运用在数字化转型的过程中。

实时低成本的真实数据采集代替了传统的人工采集上报，生产活动可以及时获得更全面的信息，大大增加适应变化的能力。这样的数字化能力在充满不确定性的未来越发重要：气候变化让耕作经验受到了巨大的挑战，新农人也不满足传统的种植方式和作物，设施农业变得像一个个植物工厂；洪水、泥石流、地震、干旱、海啸等次生灾害更加频繁，需要更多的数据以尽早预测与防御；可持续发展对工业生产和消费过程都提出了更高的要求，精确高效节制地满足需求比盲目的规模化生产更有价值。

数字化的推进也带来很多新的机会：分布式的清洁能源小心翼翼地适应外部的天气，尽量提升投入产出比，以提高竞争力；变聪明的机器从车间走向更多丰富的场景，倍增了人类的生产劳作效率；各种科学研究在更多数据样本的帮助下能够更快速地发现新规律，并且随着物联网的基础建设更快地产业化。

拿起这本书的你正处在一个有趣的时代，物联网的技术已经变得前所未有的简单，而各个行业数字化的进程才刚刚开始。这本书的源于一个非常成功的开源项目，它是 GitHub 上 Star 数最多的物联网教程之一，也是我们团队见到的最深入浅出的内容。这个教程融合了微软物联网专家多年的实践经验，可以通读理解物联网的基本逻辑，也值得一步步实践完成自己的物联网方案部署。

希望这是你学习物联网的最后一本书。止步于此，留更多的时间用越来越简单和强大的工具解决自己领域的问题。就像普通人很少再为 PC 兴奋，可见的未来我们不需要再频繁谈论 IoT 的概念，不需要从头搭建物联网系统，甚至不需要专门去研究和学习。连接与智能自然地融入各种科技与设备，我们不用为物联网欢呼，因为它已是随处可见的稀松平常的工具。

然后大家就习惯了 IoT。

潘昊
矽递科技和柴火创客空间创始人

前言

本书的绝大部分内容来自微软的 Azure 云大使提供的，一个为期 12 周的 24 课时的物联网基础知识开源课程——IoT for Beginners。此课程的每节课都包括课前准备和课后测验、知识点讲解、解决方案、作业等。课程作者们以项目为基础的教学法，使读者能够边学边做，这是一种证明新技能能够快速“落地”的有效方法。

这些项目涵盖了食物从农场到餐桌的整个过程，包括从农场、物流、制造、零售到消费者——涵盖了所有物联网设备的流行行业领域。

英文版课程创作团队

衷心感谢英文版课程的作者 Jen Fox、Jen Looper、Jim Bennett，以及插画艺术家 Nitya Narasimhan（有关 Nitya Narasimhan 的介绍请参看🔗 [L0-1]），她用插画的方式将每课的知识点做了总结（见下页插画）。

同时也感谢我们的微软学习学生大使团队，他们一直在审查和翻译这个课程，包括 Aditya Garg、Anurag Sharma、Arpita Das、Aryan Jain、Bhavesh Suneja、Faith Hunja、Lateefah Bello、Manvi Jha、Mireille Tan、Mohammad Iftekhher (Iftu) Ebne Jalal、Mohammad Zulfikar、Priyanshu Srivastav、Thanmai Gowducheruvu，以及 Zina Kamel。

👤 [L0-2] 通过链接索引的视频，接受英文版课程的作者和学生大使团队的问候。

中文翻译及编辑团队

中文翻译及编辑团队来自柴火创客空间与矽递科技，包括冯磊、黄夏、龚莉钧、黎孟度、刘海旭、潘知非（实习）、杨雨婷（实习）；版式设计及排版由孟依卉和冯磊完成。

中文版修订特别感谢

- 来自微软 Cloud Advocate 部门的 Jim Bennett 和卢建晖（中国）。
- 紫金山实验室的陈声健老师。
- 广东工程职业技术学院的许喜斌老师。
- 清华大学精密仪器系的曾悦老师。
- 南方科技大学创新创意设计学院的罗涛老师。

给教师的建议

老师们，我们已经提出了一些关于如何使用该课程的建议🔗 [L0-3]（英文）。如果你想创建自己的课程，我们也提供了一个课程模板🔗 [L0-4]（英文）。

📄 IoT-For-Beginners 开源文档

微软已将整个 IoT for Beginners（英文版）的课程在 Github 上开源，🔗 <https://github.com/microsoft/IoT-For-Beginners>。中国用户可以通过 Gitee 上的副本访问，🔗 <https://gitee.com/seeed-projects/IoT-For-Beginners>，本书的课程相关程序都以 Gitee 的链接地址给出。

🔗 关于课程内的链接编号

英文版的课程包含了大量的链接，而且有很多链接非常长，印在书上会不便使用，所以本书的大部分链接都通过链接编号提供，读者可以通过此书的链接列表页面🔗 https://tinkergen.cn/book-iot_lb，或扫描下方二维码访问此页面，并依据索引编号访问对应的链接。



给学生及读者的建议

要想自学这门课程，可以在 Github 或 Gitee 上 Fork 整个项目并自己阅读完成练习，从课前测验开始，然后阅读课程并完成项目和作业。建议尽量通过理解课程来自己创建和完成项目，而不是复制解决方案的代码；不过这些代码在每个面向项目的课程中的 /solutions 文件夹中都有；另一个建议是与朋友组成一个学习小组，一起学习这些内容。对于进一步的学习，我们推荐 Microsoft Learn  [L0-5]。

关于这个课程的视频概述，可以观看视频  [L0-6]。

课程结构

在构建本课程时，我们明确了两个教学宗旨：首先确保它是基于项目的，其次每课都有课前和课后的测验。在这个系列课程结束时，学生将会建立一个自动植物监测和灌溉系统、一个车辆跟踪器、一个检测食物质量和库存的智能工厂装置，以及一个语音控制的烹饪计时器；并将学习物联网的基本知识，包括如何编写设备代码、连接到云、分析遥测数据和在边缘设备上运行 AI。

此外，课前的低难度测验可以帮助学生明确学习主题，而课后的第二次测验则能进一步强化记忆。这个课程的设计力求灵活有趣，学生可以全部选择或选择感兴趣的部分学习。课程项目的难度循序渐进，开始时很简单，越接近尾声会变得越复杂。

本课程每个项目都是基于学生和业余爱好者可用的真实世界的硬件；每个项目都探究了具体的项目领域，并提供了相关的背景知识。要成为一个成功的开发者，需要养成了解你所要解决问题的领域知识的习惯，课程里提供的背景知识可以让读者深度思考其物联网解决方案，同时让物联网开发者在近乎实战的背景下学习。读者可以学习反思自己正在构建的解决方案的目的和意义，培养对最终用户的同理心。

硬件

我们为课程提供了两类物联网硬件，选择哪个取决于你对编程语言知识或偏好、学习目标和可用性的个人喜好。我们还提供了一个“虚拟硬件”版本，供那些没有硬件或者想在购买之前了解更多的人使用。你可以阅读更多内容，并在硬件页面上找到一个“购物清单”，包括从我们的朋友矽递科技购买完整套件的链接。

每一课都包括以下结构：

- 插图说明。
- 可选的补充视频。
- 课前测验。
- 知识点讲解。
- 基于项目的课程，关于如何建立项目的分步指南。
- 知识检查。
- 挑战。
- 复习和自学。
- 课后测验。
- 作业。

课程表

见表 0-1。

其他课程推荐

我们的团队还制作了以下课程！请读者查看。

- Web 开发入门（Web Dev for Beginners）
 [L0-7]（英文）。
- 机器学习入门（ML for Beginners）
[L0-8]（英文）。 
- 数据科学入门（Data Science for Beginners）
 [L0-9]（英文）。

图片来源

你可以在“attributions.md”  [L0-10] 中找到本课程中使用的图片的来源说明。

表 0-1 课程表

项目名	课程标题	学习目标
入门篇	第 1 课 物联网简介	了解物联网的基本原理和物联网解决方案的基本构件，如传感器和云服务，同时设置你的第一个物联网设备
	第 2 课 深入了解物联网	了解更多关于物联网系统的组成部分，以及微控制器和单板计算机
	第 3 课 通过传感器和执行器与物质世界交互	了解从物质世界收集数据的传感器，以及发送反馈的执行器，同时制作一个夜灯
	第 4 课 将你的设备连接到互联网	了解如何将物联网设备连接到互联网，通过将你的小夜灯连接到 MQTT 代理来发送和接收信息
农场篇	第 5 课 用物联网预测植物生长	了解如何利用物联网设备捕获的温度数据预测植物生长
	第 6 课 检测土壤水分	了解如何检测土壤水分和校准土壤水分传感器
	第 7 课 自动浇灌植物	了解如何使用继电器和 MQTT 自动浇灌植物并计时
	第 8 课 将你的种植数据迁移到云端	了解云和云端托管的物联网服务，以及如何将你的植物连接到其中一个，而不是公共的 MQTT 代理
	第 9 课 将你的应用逻辑迁移到云端	了解如何在云端编写响应物联网消息的应用逻辑
	第 10 课 确保你的植物安全	了解物联网的安全性，以及如何用密钥和证书保持你的植物安全
运输篇	第 11 课 位置追踪	了解物联网设备如何使用 GPS 进行位置追踪
	第 12 课 存储位置数据	了解如何存储物联网数据，以便日后进行可视化或分析
	第 13 课 可视化位置数据	了解可视化的位置数据在地图上，以及二维地图如何代表真实的三维世界
	第 14 课 地理围栏	了解地理围栏，以及如何利用它来提醒供应链中的车辆接近目的地
制造篇	第 15 课 训练水果质量检测器	了解如何在云端训练图像分类器来检测水果质量
	第 16 课 从物联网设备检查水果质量	了解如何从物联网设备使用水果质量检测器
	第 17 课 在边缘设备上运行你的水果检测器	了解在边缘的物联网设备上运行你的水果检测器
	第 18 课 从传感器触发水果质量检测	了解如何从传感器触发水果质量检测
零售篇	第 19 课 训练一个库存检测器	了解如何使用物体检测来训练一个库存检测器来计算商店里的库存
	第 20 课 从物联网设备上检查库存	了解如何使用物体检测模型从物联网设备检查库存
居家篇	第 21 课 使用物联网设备识别语音	了解如何从物联网设备中识别语音以建立一个智能计时器
	第 22 课 理解语言	学习如何理解对物联网设备用语音说的句子
	第 23 课 设置定时器并提供口头反馈	了解如何在物联网设备上设置定时器，并就定时器的设置和结束时间提供口头反馈
	第 24 课 让设备支持多种语言	学习如何支持多种语言，包括语音和智能定时器的响应

小贴士类型

本书有许多不同类型的小贴士，根据图标可以区分它们的内容类别：

：补充介绍或进一步的解释。

 **注意：** 注意事项。

：做点研究。

：名词或概念讲解。

：及时清理云账户的提示。

：云账户和定价有关的信息。

目 录

入门篇

第 1 课 物联网简介 / 2

- 1.1 什么是“物联网” / 2
- 1.2 物联网设备 / 3
- 1.3 准备好你的设备 / 9
- 1.4 物联网的应用 / 27
- 1.5 探究你身边的物联网设备 / 28

第 2 课 深入了解物联网 / 30

- 2.1 物联网应用的组成部分 / 30
- 2.2 深入了解微控制器 / 32
- 2.3 深入了解单板计算机 / 36

第 3 课 通过传感器和执行器与物质世界交互 / 40

- 3.1 什么是传感器 / 40
- 3.2 使用一个传感器 / 41
- 3.3 传感器类型 / 47
- 3.4 什么是执行器 / 49
- 3.5 使用一个执行器 / 49
- 3.6 执行器类型 / 57

第 4 课 将你的设备连接到互联网 / 60

- 4.1 通信协议 / 61
- 4.2 消息队列遥测传输协议 (MQTT) / 61
- 4.3 遥测 / 71
- 4.4 命令 / 79

农场篇

第 5 课 用物联网预测植物生长 / 86

- 5.1 数字农业 / 86
- 5.2 为什么温度对耕作很重要 / 87
- 5.3 测量环境温度 / 88
- 5.4 生长度日 (GDD) / 96
- 5.5 使用温度传感器数据计算 GDD / 96

第 6 课 检测土壤水分 / 104

- 6.1 土壤水分 / 105
- 6.2 传感器如何与物联网设备通信 / 106
- 6.3 测量土壤中的水分 / 110
- 6.4 传感器校准 / 117

第 7 课 自动浇灌植物 / 120

- 7.1 用低功率物联网设备控制高功率设备 / 120
- 7.2 控制一个继电器 / 122
- 7.3 通过 MQTT 控制你的植物 / 129
- 7.4 控制传感器和执行器的浇水周期时间 / 129
- 7.5 在你的植物控制服务器上添加对浇水周期时间的控制 / 131

第 8 课 将你的种植数据迁移到云端 / 135

- 8.1 什么是云 / 135
- 8.2 创建一个云订阅 / 136
- 8.3 云物联网服务 / 137
- 8.4 在云中创建一个物联网服务 / 138
- 8.5 与 IoT 中心进行通信 / 141

8.6 将你的设备连接到物联网服务 / 141

第 9 课 将你的应用逻辑迁移到云端 / 153

9.1 什么是 Serverless / 153

9.2 创建一个 Serverless 应用程序 / 155

9.3 创建一个 IoT 中心事件触发器 / 159

9.4 从 Serverless 代码中直接发送方法请求 / 162

9.5 将你的 Serverless 代码部署到云端 / 164

第 10 课 确保你的植物安全 / 168

10.1 为什么你需要保护物联网设备 / 168

10.2 密码学 / 169

10.3 保护你的物联网设备 / 171

10.4 生成和使用 X.509 证书 / 173

运输篇

第 11 课 位置追踪 / 178

11.1 车联网 / 179

11.2 地理空间坐标 / 179

11.3 全球定位系统 (GPS) / 181

11.4 读取 GPS 传感器的数据 / 181

11.5 NMEA 协议 GPS 数据 / 190

11.6 GPS 传感器数据解码 / 191

第 12 课 存储位置数据 / 195

12.1 结构化和非结构化的数据 / 195

12.2 将 GPS 数据发送到 IoT 中心 / 197

12.3 热、温、冷数据路径 / 198

12.4 使用 Serverless 代码处理 GPS 事件 / 199

12.5 Azure 存储账户 / 199

12.6 将你的 Serverless 代码与存储相连 / 200

第 13 课 可视化位置数据 / 206

13.1 什么是数据可视化 / 207

13.2 地图服务 / 208

13.3 创建 Azure 地图资源 / 208

13.4 在网页上显示地图 / 209

13.5 GeoJSON 格式 / 210

13.6 使用 GeoJSON 在地图上绘制 GPS 数据 / 211

第 14 课 地理围栏 / 215

14.1 什么是地理围栏 / 216

14.2 定义一个地理围栏 / 217

14.3 针对地理围栏的测试点 / 220

14.4 在 Serverless 代码中使用地理围栏 / 222

制造篇

第 15 课 训练水果质量检测器 / 228

15.1 使用人工智能和机器学习对食物进行分类 / 228

15.2 通过机器学习进行图像分类 / 229

15.3 训练一个图像分类器 / 230

15.4 自定义视觉工具 / 231

15.5 测试你的图像分类器 / 234

15.6 重新训练你的图像分类器 / 234

第 16 课 用物联网设备检查水果质量 / 236

16.1 相机传感器 / 236

16.2 使用物联网设备捕捉图像 / 237

16.3 发布你的图像分类器 / 250

16.4 对来自你的物联网设备的图像进行分类 / 252

16.5 改进模型 / 259

第 17 课 在边缘设备上运行你的水果检测器 / 261

17.1 边缘计算 / 261

17.2 Azure IoT Edge / 263

17.3 注册一个物联网边缘设备 / 264

17.4 设置一个物联网边缘设备 / 265

17.5 输出你的模型 / 267

17.6 准备好你的容器进行部署 / 270

17.7 部署你的容器 / 273

17.8 使用你的物联网边缘设备 / 277

第 18 课 从传感器触发水果质量检测 / 282

18.1 构建复杂的物联网应用 / 282

18.2 设计一个水果质量控制体系 / 284

18.3 用传感器触发水果质量检测 / 285

18.4 将数据用于水果质量检测器 / 293

18.5 使用开发者设备来模拟多个物联网设备 / 294

18.6 转向生产 / 294

零售篇

第 19 课 训练一个库存检测器 / 298

19.1 对象检测 / 298

19.2 在零售业中使用对象检测 / 300

19.3 训练一个对象检测器 / 300

19.4 测试你的对象检测器 / 302

19.5 重新训练你的对象检测器 / 303

第 20 课 从物联网设备上检查库存 / 305

20.1 库存盘点 / 305

20.2 从你的物联网设备中调用你的对象检测器 / 306

20.3 边界框 / 312

20.4 重新训练模型 / 313

20.5 计算库存 / 313

居家篇

第 21 课 使用物联网设备识别语音 / 330

21.1 麦克风 / 331

21.2 从你的物联网设备捕获音频 / 333

21.3 语音转文本技术介绍 / 355

21.4 在物联网设备上将语音转换为文本 / 356

第 22 课 理解语言 / 372

22.1 语言理解 / 372

22.2 创建一个语言理解模型 / 373

22.3 意向和实体 / 374

22.4 使用语言理解模型 / 377

第 23 课 设置定时器并提供口头反馈 / 384

23.1 文本转语音技术介绍 / 384

23.2 设置定时器 / 385

23.3 将文本转换为语音 / 393

第 24 课 让设备支持多种语言 / 409

24.1 翻译文本 / 409

24.2 翻译服务 / 411

24.3 创建一个翻译器资源 / 411

24.4 在有翻译的应用程序中支持多语言 / 412

24.5 使用人工智能服务翻译文本 / 412

附录 A 本书主页及习题答案 / 427

入门篇

在本篇中，你将学习到物联网的基本概念，并建立你的第一个连接到云的“Hello World”物联网项目。这个项目是制作一个智能小夜灯，当传感器测量到的光照度下降时，就会亮起来。



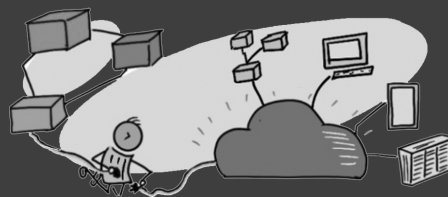
课程简介

本篇包含以下课程：

- 第 1 课 物联网简介
- 第 2 课 深入了解物联网
- 第 3 课 通过传感器和执行器与物质世界交互
- 第 4 课 将你的设备连接到互联网

感谢

本篇所有的课程都是由 Jen Looper 用♥编写。



第 1 课

物联网简介

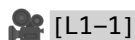


课前准备

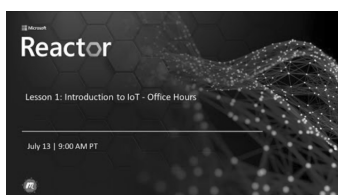
本课是微软 Reactor（微软为所有开发者及初创企业提供的社区空间）的 Hello IoT 系列课程的一部分。课程开始通过 2 个视频（英文）对物联网的基础做了介绍。



观看视频：吉姆老师讲物联网



[L1-1]



观看视频：物联网介绍——自

学时间  [L1-2]



简介

本课涵盖了一些围绕物联网基础知识的话题，并让你开始设置你的硬件。

在本课中，我们的学习内容将涵盖：

- 1.1 什么是“物联网”
- 1.2 物联网设备
- 1.3 准备好你的设备
- 1.4 物联网的应用
- 1.5 探究你身边的物联网设备

课前测验

（1）IoT（物联网）中的 I 代表（ ）。

- A. 互联网 Internet
- B. 铱星 Iridium
- C. 熨烫 Ironing

（2）到 2025 年年底，正在使用的物联网设备的估计数量有（ ）。

- A. 30 个
- B. 3000 万个
- C. 300 亿个

（3）智能手机是物联网设备。这个说法（ ）。

- A. 正确
- B. 错误

1.1 什么是“物联网”

“物联网 (Internet of Things, IoT)”一词是由凯文·阿什顿 (Kevin Ashton) 在 1999 年创造的，指的是通过传感器将互联网与物理世界连接起来。从那时起，这个词就被用于描述任何与周围物理世界互动的设备，要么从传感器收集数据，要么通过执行器（会处理一些事件的设备，如打开开关或点亮 LED）提供与物理世界的互动，一般都与其他设备或互联网连接。

 **传感器：**从物理世界收集信息，如测量速度、温度或位置。

执行器：将电信号转换为与物理世界的互动行为，如触发开关、打开灯、发出声音，或向其他硬件发送控制信号，如接通插座电源。

物联网作为一个技术领域，不仅仅包含设备，它还包括基于云的服务，它可以处理传感器数据，或向连接到物联网设备的执行器发送请求。它还包括没有或不需要互联网连接的设备，通常被称为边缘设备。这些设备可以自己处理和响应传感器数据，通常使用在云端训练的 AI 模型。

物联网是一个快速增长的技术领域。据估计，到 2025 年年底，有 300 亿台物联网设备将被部署并连接到互联网，如图 1-1 所示。展望未来，估计到 2025 年，物联网设备将收集近 80 泽字节（1 泽字节 = 10^9 TB 或 10^{12} GB）数据。这是一个很大的数据量！

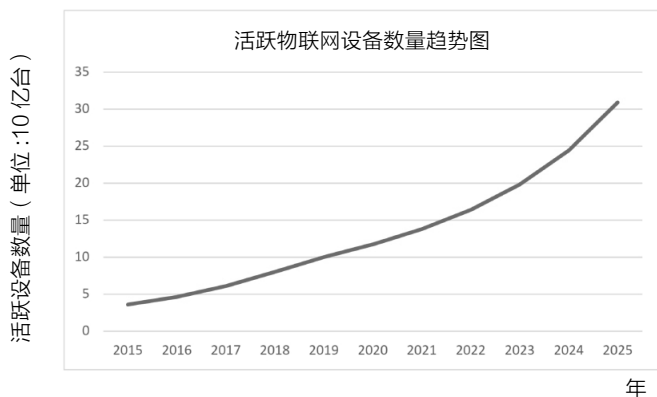


图 1-1 这是一张显示活跃的物联网设备随时间变化的图表，从 2015 年的不足 50 亿台到 2025 年的超过 300 亿台，呈上升趋势

做点研究

物联网设备产生的数据中，有多少是真正被使用的，有多少是浪费的？为什么这么多的数据被忽视？

这些数据是物联网成功的关键。要成为一个成功的物联网开发者，你要理解你所需收集的数据，以及如何收集，如何根据这些数据做出决定，以及如何在需要时利用这些决定与物理世界互动。

1.2 物联网设备

物联网中的“T”指的是“Things（物）”——通过从传感器收集数据，或通过执行器提供与现实世界、与周围物理世界互动的设备。

用于生产或商业用途的设备，如消费者健身追踪器或工业机器控制器，通常是定制的。它们使用定制的电路板，甚至可能是定制的处理器的，旨在满足特定任务的需要，无论是小到可以戴在手腕上的，还是坚固到可以在高温、高压或高振动的工厂环境中工作的。

作为一个学习物联网或创建设备原型的开发者，通常需要从开发者套件开始。这些是为开发人员设计的通用物联网设备，具有在生产设备上不会有的功能，如提供一组外部引脚连接传感器或执行器，支持可编程的硬件，或添加了在进行大规模生产时会增加不必要成本的额外资源。

这些开发套件通常分为两类——微控制器和单板计算机。这里先简单介绍一些，在第 2 课将进行更详细的介绍。

你的手机也可以被认为是一个通用的物联网设备，内置有传感器和执行器，不同的应用程序通过不同的云服务以不同的方式使用传感器和执行器。你甚至可以找到一些物联网教程，将手机应用作为物联网设备。

1.2.1 微控制器——MCU

微控制器（Micro Control Unit, MCU）是一种小型计算机，由以下部分组成。

- 一个或多个中央处理器（CPU）：微控制器的“大脑”，用来运行程序。
- 内存（RAM 和程序存储器）：存储程序、数据和变量的地方。
- 可编程输入/输出（I/O）连接：与外设（连接的设备）对话，如传感器和执行器。

在互联网上搜索微控制器时，用 MCU 作为关键字要小心，因为这将带来很多关于 Marvel Cinematic Universe（漫威电影宇宙）的结果，而不是微控制器。

微控制器通常是用 C/C++ 编程的。

微控制器是典型的低成本计算设备，用于定制硬件的微控制器的平均价格已下降到几元人民币，有些设备甚至便宜到几角钱。有些开发者工具包甚至可以低至 20 元左右，随着更多功能的添加，成本也会上升。Wio Terminal 是矽递科技的一个基于微控制器的开发工具包，这个小盒子还集成了传感器、执行器、Wi-Fi 和一个屏幕，价格为 280 元人民币左右，如图 1-2 所示。

微控制器被设计为通过编程来完成有限且非常具体的任务，而不是像 PC 那样的通用计算机。除了非常特殊的情况，你不能为微控制器连接显示器、键盘和鼠标，并将其用于通用任务。

微控制器开发板通常带有额外的板上传感器和执行器。大多数板子会有一个或多个可以编程的 LED 以及其他设备，如标准接口，用于添加更多的传感器或执行器，使用各种制造商的生态系统或内置传感器（通常是最流行的，如温度传感器）。一些微控制器有内置的无线连接，如蓝牙或 Wi-Fi，或在板上有额外的微控制器来增加这种连接。



图 1-2 Wio Terminal

1.2.2 单板计算机——SBC

单板计算机（Single-Board Computer, SBC）是一种小型计算设备，它将完整计算机的所有元素都包含在一块小板上。这些设备的规格接近台式机或 PC，运行完整的操作系统，但体积小，耗电少，而且价格要便宜得多，如图 1-3 所示的树莓派。

像微控制器一样，单板计算机有一个 CPU、内存和输入 / 输出引脚，但它们有额外的功能，如图形芯片，允许你连接显示器、音频输出和 USB 端口，以连接键盘鼠标和其他标准 USB 设备，如网络摄像头或外部存储等。程序与操作系统一起存储在 SD 卡或硬盘上，而不是内置在电路板上的内存芯片里。

单板计算机是功能齐全的计算机，所以可以用任何语言进行编程。物联网设备通常是用 Python 编程的。

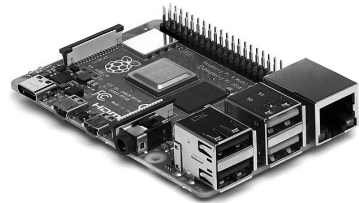


图 1-3 第 4 代树莓派，树莓派是最受欢迎的单板计算机之一

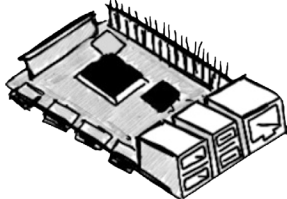
你可以把单板计算机看作比你常用的 PC 更小、更便宜的版本，并增加了 GPIO（通用输入 / 输出）引脚，以便与传感器和执行器互动。

1.2.3 课程的硬件选择

后续所有课程都包括使用物联网设备与物理世界互动并与云端通信的作业。每节课都提供了三种设备以供选择，如图 1-4 所示。



Arduino：使用矽递科技的 Wio Terminal



单板计算机：第 4 代树莓派



单板计算机：在你的 PC 上运行的虚拟单板计算机

图 1-4 每节课都提供了三种设备选择

可以在下页的“连接设备：硬件指南”中了解完成所有作业所需的硬件。

你甚至不需要购买任何物联网硬件来完成作业，你可以使用虚拟单板计算机来完成一切操作。

连接设备：硬件指南



IoT（物联网）中的“T”代表“物（Things）”，是指与我们周围世界互动的设备。每个项目都是基于学生和电子爱好者可用的真实世界的硬件。我们有两种物联网硬件可供选择，取决于个人喜好、编程语言知识或偏好、学习目标和可用性。我们还提供了一个“虚拟硬件”版本，供那些没有机会获得硬件，或想在购买硬件之前了解更多信息的人使用。

本课程中物理硬件选择 Arduino 或树莓派。每种硬件都有自己的优点和缺点，这些在 1.2.3 节中都有涉及。如果你还没有选定一个硬件平台，你可以看看第 2 课来选定你最感兴趣的硬件平台。

选择特定的硬件是为了减少课程和作业的复杂性。尽管其他硬件也可以使用，但我们不能保证所有的作业都能在你的设备上得到支持而不需要额外的硬件。例如，很多 Arduino 设备没有 Wi-Fi，而连接到云端需要 Wi-Fi。选择 Wio Terminal 是因为它有内置的 Wi-Fi。

你还需要一些非技术性的物品，如土壤或盆栽，以及水果或蔬菜等。

📌 你不需要购买任何物联网硬件来完成作业。你可以使用虚拟物联网硬件来完成一切。

需要的工具包

Seeed Studio（矽递科技）的标志，矽递科技非常友好地将本课程所需的大部分硬件作为易于购买的套件进行提供。

seeed studio



Arduino: SenseCAP K1101 – 物联网原型开发套件

矽递科技自研的 SenseCAP K1101 – 物联网原型开发套件  [L1-3]，可提供本课程所需的大部分模块，如图 1-5 所示。Arduino 的所有设备代码都是用 C++ 语言。为了完成所有的作业，你将需要以下东西。

（1）Arduino 硬件。

- Wio Terminal。
- 可选的 USB-C 电缆或 USB-A 转 USB-C 的适配器。Wio Terminal 有一个 USB-C 端口，并配有 USB-C 转 USB-A 的电缆。如果你的 PC 只有 USB-C 端口，你将需要一条 USB-C 电缆，或一个 USB-A 转 USB-C 的适配器。

（2）Arduino 的传感器和执行器。

以下是使用 Wio Terminal 的 Arduino 专用设备，与使用树莓派的无关。

- ArduCam Mini 2MP Plus – OV2640 摄像头模块（不在套件内，见图 1-6， [L1-3-2]）。
- Grove – Vision AI Module with Himax HX6537 摄像头模块。
- Grove 温湿度传感器模块（SHT40）。
- ReSpeaker 2-Mics Pi HAT 双麦克风扩展板（不在套件内， [L1-4]）。
- 面包板跳线（不在套件内，见图 1-6）。
- 耳机或其他带有 3.5 毫米插孔的扬声器，或一个 JST 扬声器，如 Mono Enclosed Speaker —— 2 瓦 6 欧姆（不在套件内，见图 1-6， [L1-5]）。
- 16GB 或容量更小的 microSD 卡（不在套件内，见图 1-6），如果你没有内置的 SD 卡，则还需要一个连接器将其用于你的计算机。

⚠ 注意

Wio Terminal 只支持 16GB 以下的 SD 卡，不支持更大的容量。

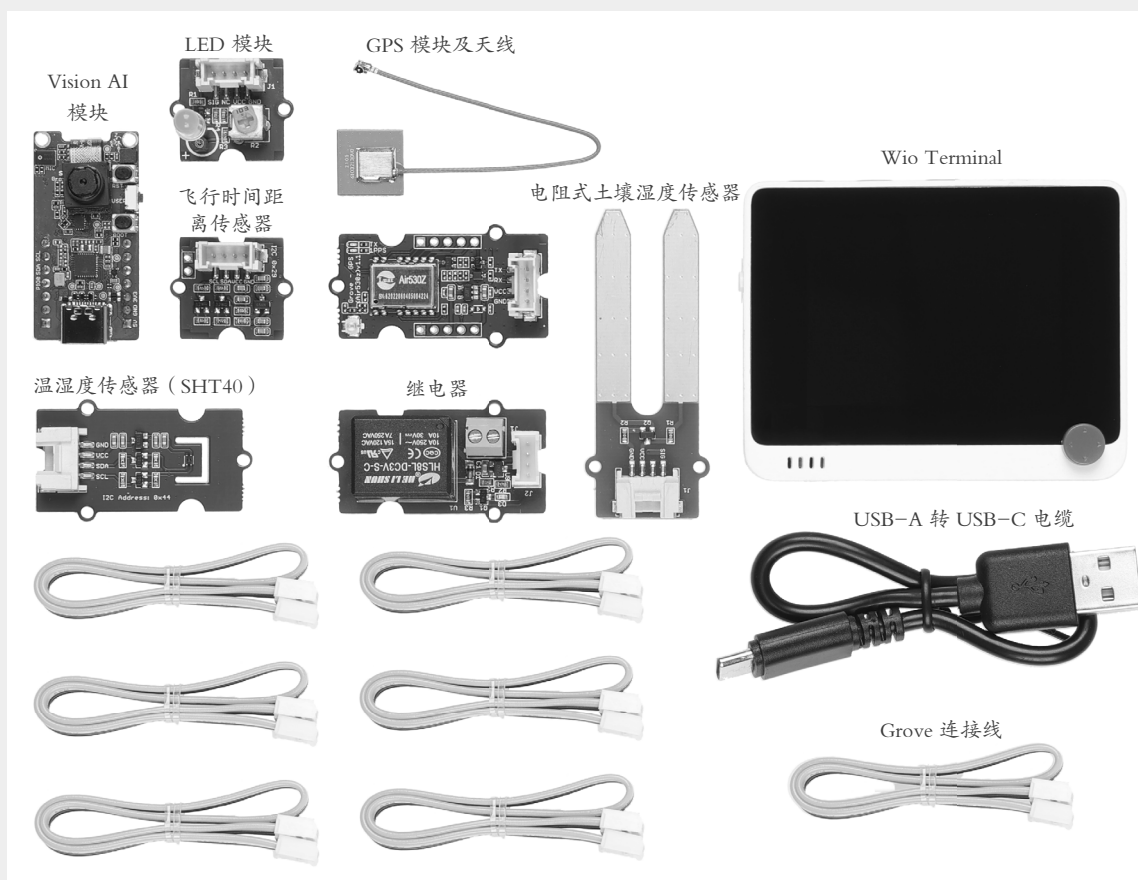


图 1-5 Arduino : 基于 Wio Terminal 的 SenseCAP K1101 – 物联网原型开发套件

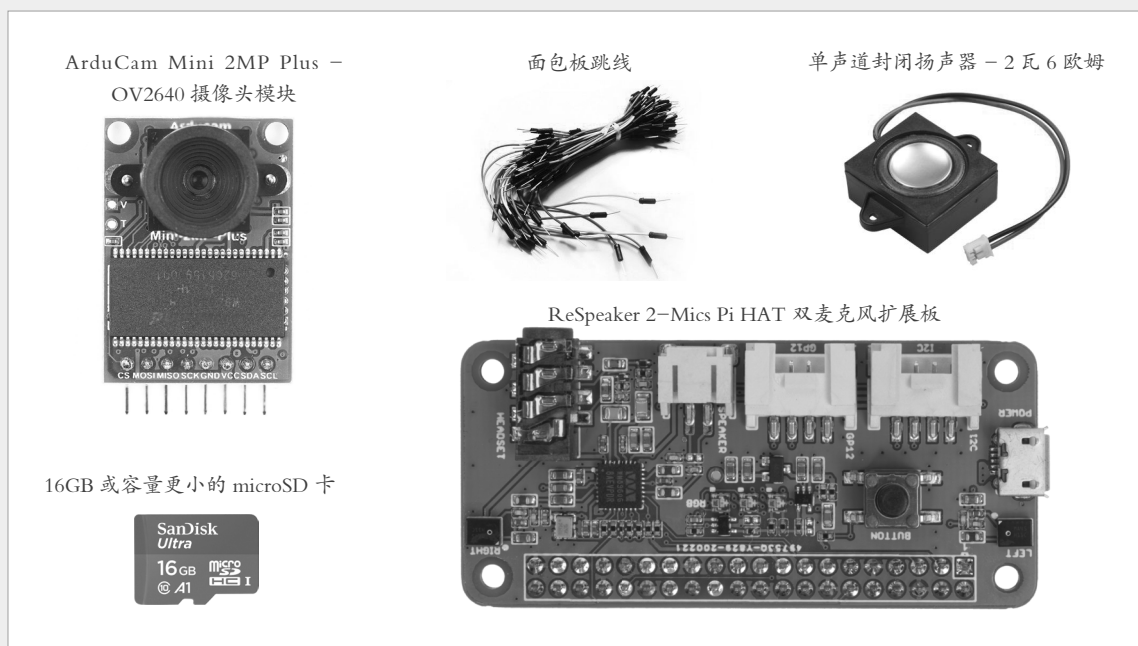


图 1-6 Arduino : 非套件范围内的物品



树莓派：物联网初学者—树莓派 4 入门套件

矽递科技提供的物联网初学者—树莓派 4 入门套件 [L1-6]，如图 1-7 所示。树莓派的所有设备代码都是 Python 语言。要完成所有的作业，你将需要以下条件。

(1) 树莓派硬件。

- 树莓派 4。
- microSD 卡（你可以买到带有 microSD 卡的树莓派套件），如果你没有内置的 SD 卡，还需要一个读卡器，以便在你的计算机上使用，建议使用 32GB 及更大容量的卡，如图 1-8 所示。
- USB 电源（你可以买到带有电源的 Raspberry Pi 4 套件）。如果你使用的是 Raspberry Pi 4，你需要一个 USB-C 接口电源，早期版本的设备需要一个 micro-USB 接口电源。

(2) 树莓派专用传感器和执行器。

- Grove Pi base HAT 树莓派的 Grove 扩展板。
- Raspberry Pi Camera module 树莓派相机模块。
- Grove 温湿度传感器模块（DHT11）。
- 麦克风和扬声器，使用以下其中一个（或同等的）：如果你的树莓派与带有扬声器的显示器或电视相连，可以使用任何 USB 麦克风与任何 USB 扬声器，或带有 3.5 毫米插孔的扬声器，或使用 HDMI 音频输出或任何带有内置麦克风的 USB 耳机。
- ReSpeaker 2-Mics Pi HAT 配备了耳机或其他带有 3.5 毫米插孔的扬声器，或一个 JST 扬声器，如单声道封闭式扬声器 - 2 瓦 6 欧姆（见图 1-8），或 USB 免提电话。
- Grove 光线传感器模块。
- Grove 按钮模块。

⚠ 注意

树莓派 2B 及以上的版本应该可以完成这些课程中的作业。如果你打算在树莓派上直接运行 VS Code（微软的编程工具），那么就需要一个有 2GB 或更大存储空间的 4 代树莓派版本。如果你打算远程访问树莓派，那么任何 2B 及以上版本的树莓派都可以使用。

树莓派专用传感器和执行器与使用 Arduino 设备无关。

传感器和执行器

大部分所需的传感器和执行器在 Arduino 和树莓派的学习路径上都有使用：

- Grove LED。
- Grove GPS (Air530) GPS 模块。
- Grove Relay 继电器模块。
- Grove 飞行时间距离传感器模块。
- Grove 电阻式土壤湿度传感器模块。

可选硬件

关于自动浇水的课程会使用一个继电器工作。作为一种选择，你可以使用下面列出的硬件将这个继电器连接到一个由 USB 供电的水泵上。

- 6V 水泵。
- 红色和黑色的电线。
- USB-A 公插头转 5 针接线端子。
- 小平头螺丝刀。
- 硅胶管。

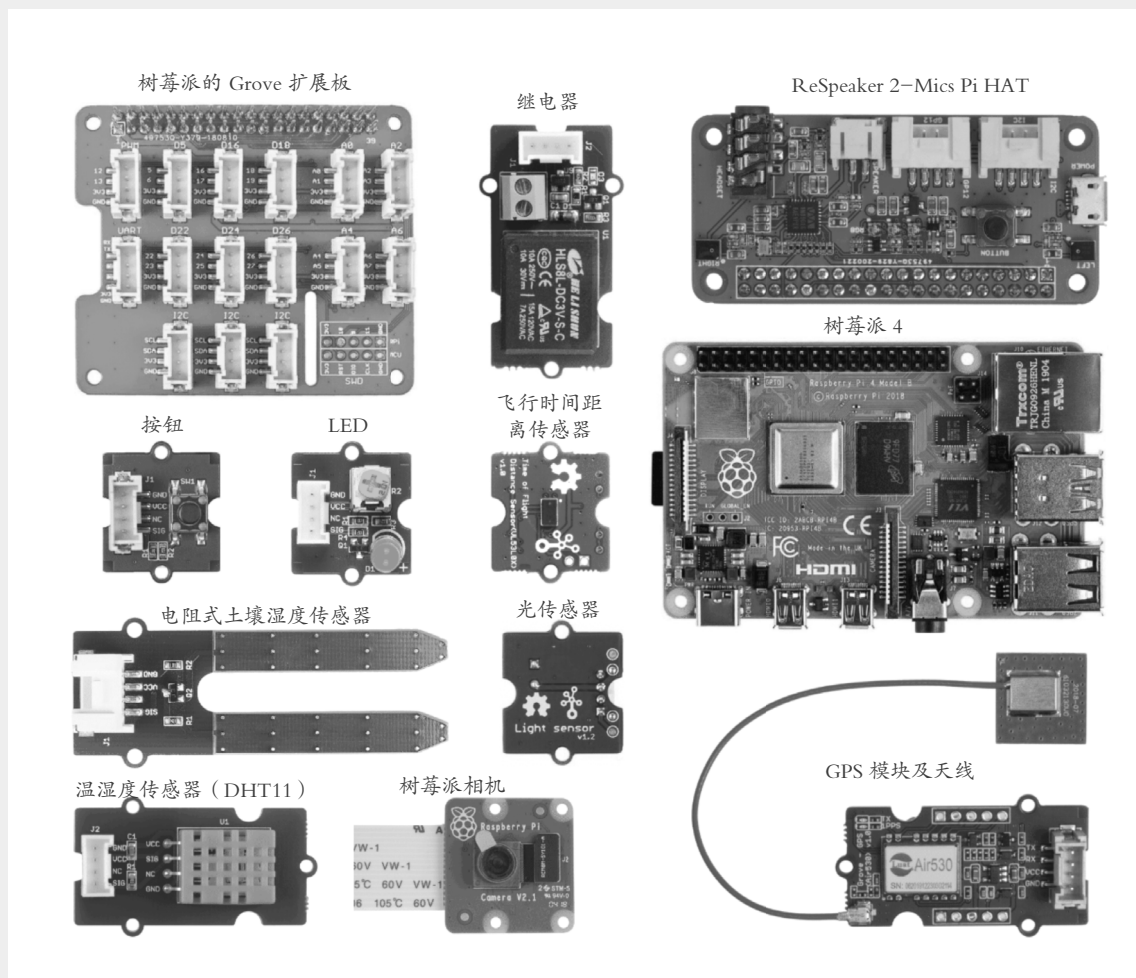


图 1-7 树莓派：矽递和微软合作的物联网初学者——树莓派 4 入门套件



图 1-8 树莓派：非套件范围内的物品

虚拟硬件

走虚拟硬件路线会使用基于 Python 实现的传感器和执行器的模拟器。根据你的硬件可用性，你可以在你常用的开发设备上运行，如 PC 或者树莓派，可以只模拟你没有的硬件。例如，如果你有树莓派摄像头，但没有 Grove 传感器，你就可以在你的树莓派上运行虚拟设备的代码，模拟 Grove 传感器。

虚拟硬件将使用 CounterFit 项目（CounterFit 是一种工具，旨在虚拟各种物联网硬件组件，如 LED、按钮、温度传感器等，然后你可以从计算机上运行物联网设备代码而不是在物联网设备上访问这些硬件组件）。

为了完成这些课程，你将需要有一个网络摄像头、麦克风和音频输出设备，如扬声器或耳机。这些设备可以是内置的，也可以是外置的，需要配置成在你的操作系统中可以工作，并可在所有应用程序中使用。

再次说明，你不需要购买任何物联网硬件来完成作业。你可以使用虚拟物联网硬件来完成一切。

硬件的选择取决于你——比如根据家里或学校已有的硬件，以及你已掌握的或打算学习的编程语言。单板计算机和虚拟单板计算机将使用相同的传感器方案，因此，如果你选择了其中一种开始学习，你可以切换到另一种硬件上而不用更换大部分的套件。在虚拟单板计算机上学习和在树莓派上学习一样，如果你最终拥有一套树莓派和传感器硬件，则大部分的代码是可以转移到树莓派上的。

1.2.4 Arduino 开发工具包

如果你对学习微控制器开发感兴趣，你可以使用 Arduino 设备完成作业。你将需要对 C/C++ 编程有基本的了解，因为课程将只教授与 Arduino 框架、正在使用的传感器和执行器以及与云端互动的库相关的代码。

作业将使用微软的 Visual Studio Code（以下简称 VS Code，标志见图 1-9）与 PlatformIO 扩展（标志见图 1-10）进行微控制器开发。如果你对 Arduino IDE 有经验，你也可以使用这个工具，但本书不会对其提供说明。



图 1-9 Visual Studio Code 的标志



图 1-10 PlatformIO 的标志

1.2.5 单板计算机开发套件

如果你对学习使用单板计算机的物联网开发感兴趣，你可以使用树莓派，或在你 PC 上运行的虚拟设备来完成作业。

你需要对 Python 编程有基本的了解，因为课程只教授与正在使用的传感器和执行器有关的代码，以及与云端互动的库。

课程作业将使用 VS Code。

如果你使用的是树莓派，你可以使用树莓派的操作系统（Raspberry Pi OS）在树莓派上使用 VS Code 进行所有的编程，或者将你的树莓派作为一个没有外设的无头设备运行，并使用 VS Code 的远程 SSH 扩展在你的 PC 上进行编程（编程效率更高），该扩展允许你连接到你的树莓派并编辑、调试和运行代码，就像你直接在设备上进行编程一样。

如果你使用虚拟设备选项，你将直接在你的计算机上编码，而不是访问传感器和执行器，你将使用一个工具来模拟这些硬件，提供你可以定义的传感器值，并在屏幕上显示执行器的结果。

1.3 准备好你的设备

在你开始对物联网设备进行编程之前，你需要进行少量的设置。根据你要使用的设备，按照下面的“连

接设备：运行 Hello World”的知识进行操作。

这些说明中确实会包含你会使用到的硬件或工具设计者的第三方网站链接。这是为了确保你一直在使用各种工具和硬件的最新版本。

通过相关指南来设置你的设备，并完成一个“Hello World”项目。这是为入门篇 4 个课程中的“创建一个物联网小夜灯”所迈出的第一步。

- Wio Terminal 编程之 Hello World。
- 树莓派编程之 Hello World。
- 虚拟设备编程之 Hello World。

下面你将用 VS Code 为 Arduino 和单板计算机编程。如果你以前没有使用过，请在 VS Code 网站：<https://code.visualstudio.com/> 上阅读更多信息。

注意

如果你还没有设备，请参考“连接设备：硬件指南”来帮助你决定要使用哪种设备，以及你需要购买哪些额外的硬件。你也可以不购买硬件，因为所有的项目都可以在虚拟硬件上运行。

连接设备：运行 Hello World



Wio Terminal 编程之 Hello World

矽递科技的 Wio Terminal（图 1-11）有一个与 Arduino 兼容的微控制器，内置 Wi-Fi 和一些传感器以及执行器，还有一些端口可以添加更多的传感器和执行器，它们都符合一个名为 Grove 的硬件生态系统规范。

设置

要使用你的 Wio Terminal，你需要在计算机上安装一些免费软件。你还需要更新 Wio Terminal 的固件，然后才能将其连接到 Wi-Fi。

任务：设置

安装所需软件并更新固件。

（1）安装 Visual Studio Code（VS Code）。这是你将使用的编辑器，用 C/C++ 编写设备代码。关于安装 VS Code 的说明，请访问 <https://code.visualstudio.com/> 参考 VS Code 教程。

（2）安装 VS Code PlatformIO 扩展。这是 VS Code 的一个扩展，支持用 C/C++ 编程微控制器。请参考 PlatformIO 扩展文档：[\[L1-7\]](#) 了解在 VS Code 中安装这个扩展的说明。这个扩展依赖于微软的 C/C++ 扩展来处理 C 和 C++ 代码，当你安装 PlatformIO 时，C/C++ 扩展会自动安装。

（3）将 Wio Terminal 连接到计算机上。Wio Terminal 的底部有一个 USB-C 接口，这需要连接到你计算机上的 USB 接口。Wio Terminal 附带一条 USB-C 转 USB-A 的电缆，但如果你的计算机只有 USB-C 接口，那么你将需要一条 USB-C 电缆或一个 USB-A 转 USB-C 的适配器。



图 1-11 一个矽递科技的 Wio Terminal

注意

另一个用于 Arduino 开发的流行 IDE 是 Arduino IDE。如果你已经熟悉这个工具，那么你可以用它来代替 VS Code 和 PlatformIO，但本课将只给出基于使用 VS Code 的说明。

注意

如果你的 Wio Terminal 是在 2022 年 7 月 15 日前购入的，则需要更新 Wio Terminal 固件，链接为：

[\[L1-8\]](#)。

编写“Hello world”程序

在开始使用一种新的编程语言或技术时，传统的做法是创建一个“Hello World”应用程序——一个输出类似“Hello World”文字的小应用程序，以显示所有工具都已正确配置。

Wio Terminal 的 Hello World 应用程序将确保你的 Visual Studio Code 与 PlatformIO 都正确安装，并对微控制器开发进行设置。

任务：创建一个 PlatformIO 项目

第一步是使用为 Wio Terminal 配置的 PlatformIO 创建一个新项目。

- (1) 将 Wio Terminal 连接到你的计算机。
- (2) 启动 VS Code。
- (3) PlatformIO 图标将出现在侧面的菜单栏上，在右侧出现的欢迎屏幕上，单击 **+ New Project** 按钮，如图 1-12 所示。

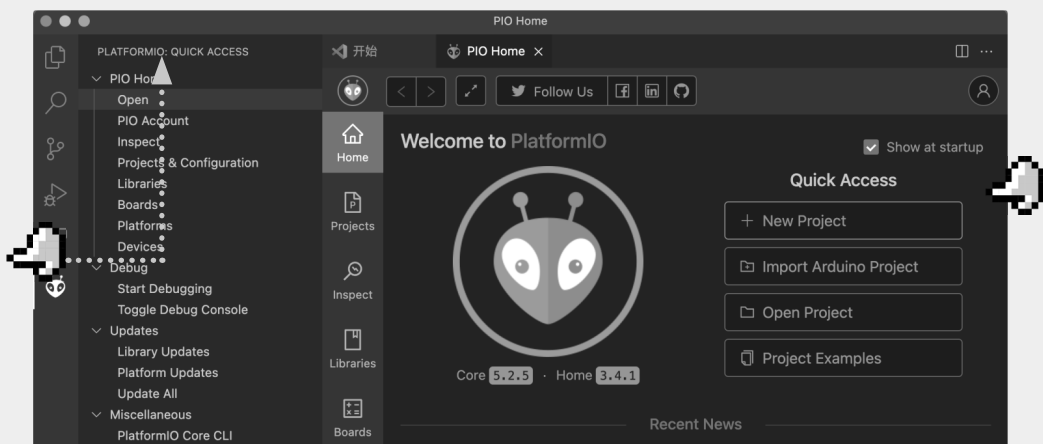


图 1-12 在左侧 PlatformIO 的图标入口单击所示的图标，在弹出的菜单 PIO Home 中单击 Open，然后在欢迎屏幕上选择 **+ New Project**

- (4) 在项目向导中配置该项目，如图 1-13 所示。

- ① 将项目命名为 **nightlight**（夜灯）。
- ② 在 Board（板子）右边的输入框中，输入 Wio 来过滤所需的板子，并选择 **Seeeduino Wio Terminal**。
- ③ 保留默认的框架（Framework）为 **Arduino Framework**。
- ④ 选中 Use default location（使用默认位置）复选框，或者取消勾选，为你的项目选择一个位置。
- ⑤ 单击 **Finish**（完成）按钮。

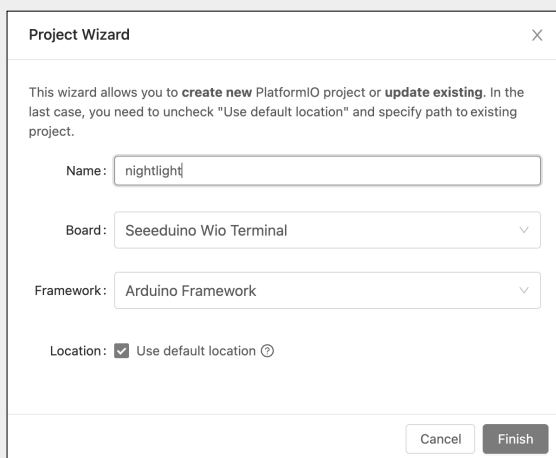


图 1-13 完成后的项目向导

PlatformIO 将下载它所需要的用来编译 Wio Terminal 的代码组件，并创建你的项目。这个过程可能需要几分钟的时间。

任务：检视 PlatformIO 项目

VS Code 的资源管理器将显示一些由 PlatformIO 向导创建的文件和文件夹，如图 1-14 所示。

文件夹介绍

- **pio/build**: 这个文件夹包含 PlatformIO 需要的临时数据，如库或编译的代码。如果被删除，它会自动重新创建，如果你要在 GitHub 等网站上分享你的项目，你并不需要把它添加到源代码版本控制中。
- **vscode**: 这个文件夹包含 PlatformIO 和 VS Code 使用的配置。如果被删除，它会自动重新创建，如果你要在 GitHub 等网站上分享你的项目，你也不需要将其添加到源代码版本控制中。
- **include**: 这个文件夹是在向你的代码添加额外的库时需要的外部头文件。在这些课程中，你不会使用到这个文件夹。
- **lib**: 这个文件夹用于存放你想在代码中调用的外部库。你不会在这些课程中使用到这个文件夹。
- **src**: 这个文件夹包含你应用程序的主要源代码。最初，它只包含一个文件，即 main.cpp。
- **test**: 这个文件夹是为代码放置任何单元测试的地方。

文件介绍

- **main.cpp**: 在 src 文件夹中的这个文件包含了你应用程序的入口点。打开这个文件，它将包含右边代码。
- 当设备启动时，Arduino 框架将运行一次 setup (设置) 函数，然后重复运行 loop (循环) 函数，直到设备关闭。
- **.gitignore**: 这个文件列出了将你的代码添加到诸如 GitHub 或 Gitee 仓库等 git 源代码版本控制时要忽略的文件和目录。
 - **platformio.ini**: 这个文件包含你的设备和应用程序的配置。打开这个文件，它将包含右边代码。
 - **[env:seed_wio_terminal]** 部分是对 Wio Terminal 的配置。可以有多个 env 部分，这样你的代码就可以为多个板子所编译了。

其他的值与项目向导的配置一致。

- **platform = atmelsam** 定义了 Wio Terminal 使用的硬件 (一个基于 ATSAMD51 的微控制器)。
- **board = seed_wio_terminal** 定义了微控制器板的类型 (Wio Terminal)。

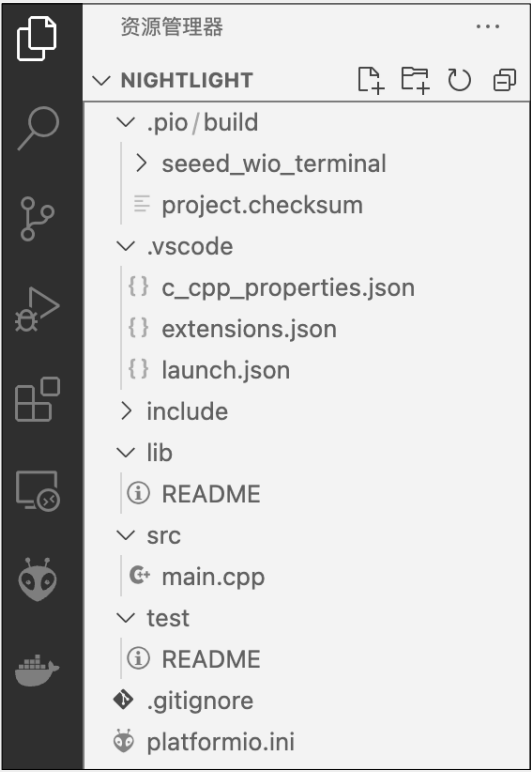


图 1-14 由 PlatformIO 向导创建的文件和文件夹

```
#include<Arduino.h>

void setup() {
    // 把你的设置代码放在这里，运行一次
}

void loop() {
    // 把你的主代码放在这里，重复运行
}
```

```
[env:seed_wio_terminal]
platform = atmelsam
board = seed_wio_terminal
framework = arduino
```

- **framework = arduino** 定义了这个项目使用 Arduino 框架。

任务：编写 Hello World 应用程序

编写 Hello World 应用程序。

(1) 在 VS Code 中打开 main.cpp 文件。

(2) 修改代码，使之与右侧的内容一致。

- ① **setup** 函数初始化了与计算机串口的连接——在本例中是用于连接 Wio Terminal 和计算机的 USB 端口。参数 9600 是波特率（也被称为传符号率），或者说是通过串口发送数据的速度，单位是每秒比特。这一设置意味着串口可以每秒发送 9600 比特（0 和 1）的数据。然后，它等待串口准备好。

- ② **loop** 函数向串口发送 **Hello World** 并换行。然后它休眠 5,000 毫秒（5 秒）。循环结束后，再运行一次，再运行一次，如此反复，直到微控制器断电为止。

(3) 让你的 Wio Terminal 进入上传模式。每次你向设备上传新的代码时，你都需要这样操作。

- ① 迅速拉下两次电源开关，每次都会弹回到开的位置，如图 1-15 所示。
- ② 检查 USB 端口右侧的蓝色状态 LED 灯，它应该是缓慢闪烁的。

(4) 建立并上传代码到 Wio Terminal。

- ① 打开 VS Code 菜单栏的“查看→命令面板”。

- ② 输入 **PlatformIO Upload** 来搜索上传选项，并选择“PlatformIO: Upload”，如图 1-16 所示。

```
#include <Arduino.h>

void setup()
{
    Serial.begin(9600);

    while (!Serial)
        ; // 等待 Serial 准备好

    delay(1000);
}

void loop()
{
    Serial.println("Hello World");
    delay(5000);
}
```

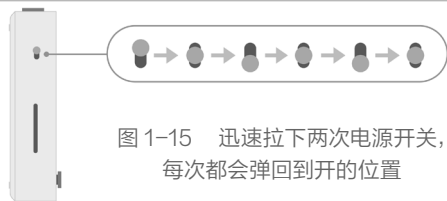


图 1-15 迅速拉下两次电源开关，每次都会弹回到开的位置

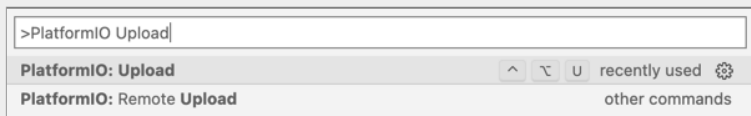


图 1-16 命令面板中的 PlatformIO Upload 选项

如果需要，PlatformIO 会在上传前自动构建代码。

- ③ 代码将被编译并上传至 Wio Terminal。

🐼 如果你使用的是 macOS，会出现一个关于 DISK NOT EJECTED PROPERLY 的通知。这是因为 Wio Terminal 作为闪存过程的一部分被挂载为一个驱动器，当编译后的代码被写入设备时，它就被断开了。你可以忽略这个通知。

PlatformIO 有一个串口监测器，可以监测从 Wio Terminal 通过 USB 电缆发送的数据。这个监测器允许你监控 **Serial.println("Hello World");** 命令发送的数据。

(1) 打开 VS Code 菜单栏的“查看→命令面板”。

(2) 在弹出的命令面板输入框内输入 **PlatformIO Serial** 来搜索串行监控选项，并选择“PlatformIO: Serial Monitor”，如图 1-17 所示。

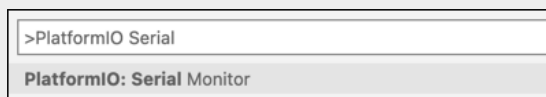


图 1-17 命令面板中的 PlatformIO Upload 选项

一个新的终端将被打开，通过串口发送的数据（Hello World）将在这个终端被打印。具体如下：

```
> Executing task: platformio device monitor <

--- Available filters and text transformations: colorize, debug, default, direct, hexlify,
log2file, nocontrol, printable, send_on_enter, time
--- More details at http://bit.ly/pio-monitor-filters
--- Miniterm on /dev/cu.usbmodem101 9600,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
Hello World
Hello World
```

Hello World 将每隔 5 秒被打印到串行显示终端上一次。在 PlatformIO 底部的导航栏中也可以找到上传和开启终端的快捷入口，如图 1-18 所示。



图 1-18 PlatformIO 底部的导航栏也可以找到上传和开启终端的快捷入口

你可以在 `code/wio-terminal` 文件夹 [\[L1-9\]](#) 中找到完整代码。

🎉 恭喜，你的“Hello World”程序创建成功了！

树莓派编程之 Hello World

树莓派是一个单板计算机，如图 1-19 所示。你可以在一个很广的硬件设备和生态系统范围内进行选择，为它添加传感器和执行器，如在本系列课程中使用的一种叫做 Grove 的硬件生态系统。你将为你的树莓派编码并使用 Python 访问 Grove 传感器。

设置

如果使用树莓派作为你的物联网硬件，那么你有两个选择——可以实践所有这些课程并直接在树莓派上进行编码，或者可以远程连接到“无头”模式的树莓派并在你的计算机上进行编码。

在你开始之前，你还需要将 Grove Base HAT 扩展板（见图 1-20）连接到你的树莓派（见图 1-21）上。

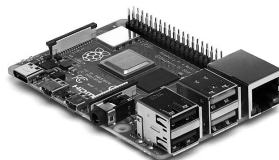


图 1-19 一台树莓派 4

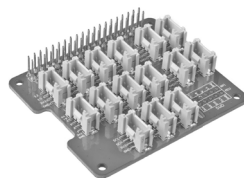


图 1-20 一块 Grove Base HAT 扩展板

任务：设置

在你的树莓派上安装 Grove Base HAT 扩展板并配置树莓派。

(1) 将 Grove Base HAT 扩展板连接到你的树莓派上。扩展板上的插座正好可以套在树莓派上的所有 GPIO 引脚上，沿着引脚方向将其按下即可牢牢地插在树莓派上。完成后扩展板就盖在了树莓派之上，如图 1-21 所示。

(2) 你需要决定如何为你的树莓派编程，并前往下面的相关部分。

- ① 直接在你的树莓派上工作。
- ② 通过远程访问在树莓派上编码。

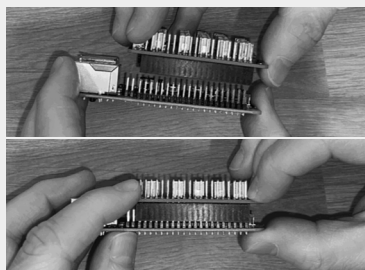


图 1-21 给树莓派套上 Grove Base HAT 扩展板

直接在你的树莓派上工作

如果你想直接在树莓派上进行编程工作，则可以使用桌面版的 Raspberry Pi OS，并安装你需要的所有工具。

任务：设置用于编程开发的树莓派

设置你用于编程开发的树莓派。

按照树莓派设置指南 [\[L1-9-2\]](#) 中的说明来设置树莓派，将键盘、鼠标、显示器连接到树莓派上，然后将它通过 Wi-Fi 或网线联网，并更新软件。

为了使用 Grove 传感器和执行器并对树莓派进行编程，你需要安装一个编辑器，以允许你编写设备代码，以及与 Grove 硬件互动的各种库和工具。

(1) 一旦你的树莓派重新启动，便可以通过单击顶部菜单栏上的终端图标来启动终端，或者选择 **菜单→附件→终端命令**。

(2) 运行以下命令以确保操作系统和安装的软件是最新的。

```
sudo apt update && sudo apt full-upgrade --yes
```

(3) 运行下面的命令，为 Grove 硬件安装所有需要的库。

```
curl -sL https://gitee.com/seeed-projects/grove.py/raw/master/install.sh | sudo bash -s -
```

Python 的强大功能之一是能够安装 Pip packages [\[L1-11\]](#)——这是由其他人编写并发布到互联网上的代码包。你可以用一个命令将 Pip 包安装到你的计算机上，然后在你的代码中使用该包。这个 Grove 安装脚本将会安装 Pip 包，你将使用 Python 与 Grove 硬件一起工作。

 默认情况下，当你安装一个包时，它在你的计算机上各个应用程序中都是可用的，这可能会导致包的版本问题——比如一个应用程序依赖于一个包的版本，而当你为不同的应用程序安装一个新的版本时，这个包就会失效。为了解决这个问题，你可以使用一个 Python 虚拟环境 [\[L1-12\]](#)，基本上是在一个专门的文件夹中的 Python 副本，当你安装 Pip 包时，它们就会被安装到这个文件夹中。在使用你的树莓派时，你将不会使用虚拟环境。Grove 安装脚本会在全局范围内安装 Grove Python 软件包。所以，要使用虚拟环境，你需要建立一个虚拟环境，然后在该环境中手动重新安装 Grove 软件包。使用全局软件包会更简单，尤其是对于很多树莓派开发者会为每个项目新刷一张干净的 SD 卡的情况。

(4) 使用菜单或在终端运行以下命令来重新启动树莓派。

注意

目前，最新的树莓派操作系统不再支持通过 PiCamera（树莓派摄像头）或任何其他 Python 库来访问相机。你需要从这里 [\[L1-10\]](#) 下载 Buster image 来安装一个旧的操作系统。

```
sudo reboot
```

(5) 一旦树莓派重新启动, 便可重新启动终端并运行以下命令来安装 Visual Studio Code (VS Code), 这是你用来编写 Python 设备代码的编辑器。

```
sudo apt install code
```

一旦安装完毕, VS Code 即可从顶部菜单中获得。

(6) 安装 Pylance。这是 VS Code 的一个扩展, 提供 Python 语言支持。请参考 VS Code 的 Pylance 扩展文档 [\[L1-13\]](#), 了解在 VS Code 中安装该扩展的说明。

如果你有喜欢的工具, 你可以在这些课程中自由使用任何 Python IDE 或编辑器, 但本书的课程将基于使用 VS Code 的情况给出教程。

通过远程访问在树莓派上编码

你还可以通过“无头”模式运行, 即不连接键盘、鼠标、显示器, 在你的计算机上使用 Visual Studio Code 对其进行配置和编码。

设置 Raspberry Pi OS

为了进行远程编码, Raspberry Pi OS 需要安装在 SD 卡上。

任务: 设置 Raspberry Pi OS

本任务目标是设置好无头的 Raspberry Pi OS。

(1) 在计算机上从 Raspberry Pi OS 软件页面 <https://www.raspberrypi.com/software/> 下载 Raspberry Pi Imager 软件并安装它。

(2) 将 SD 卡插入计算机, 必要时使用适配器。

(3) 在计算机上启动刚安装的 Raspberry Pi Imager 软件。

(4) 在 Raspberry Pi Imager 软件中, 单击 **CHOOSE OS** 按钮, 然后选择 **Raspberry Pi OS (other)**, 接着选择 **Raspberry Pi OS Lite (32-bit)**, 如图 1-22 所示。

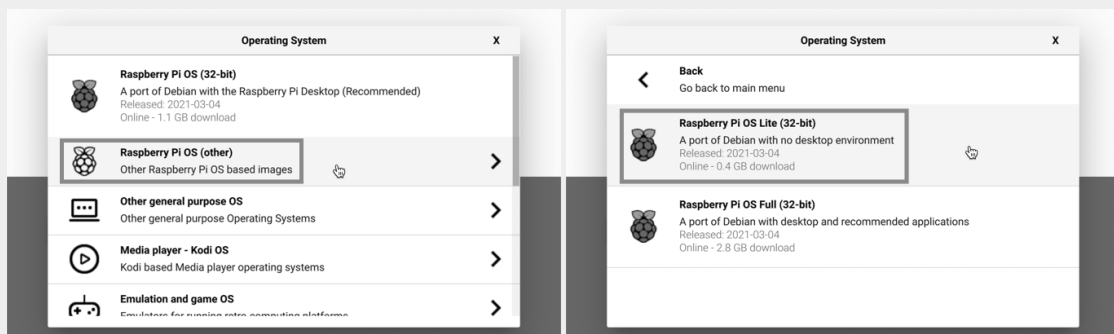


图 1-22 选择 Raspberry Pi OS Lite 后的 Raspberry Pi Imager 界面

(5) 单击 **CHOOSE STORAGE** 按钮, 然后选择你的 SD 卡。

(6) 按 Ctrl+Shift+X 组合键来启动 **Advanced Options** (高级选项)。这些选项允许在树莓派操作系统被映像到 SD 卡之前对其进行一些预配置。

① 勾选 **Enable SSH** (启用 SSH) 复选框, 并为树莓派用户设置一个密码。这是你以后用来登录树莓派的密码。

② 如果你打算通过 Wi-Fi 连接树莓派, 请勾选 **Configure**

Raspberry Pi OS Lite 是 Raspberry Pi OS 的一个版本, 没有桌面 UI 或基于 UI 的工具。对于无头的树莓派来说, 这些都是不需要的, 而且可以使安装空间更小, 启动时间更短。

Wi-Fi（配置 Wi-Fi）复选框，并输入你的 Wi-Fi SSID 和密码，还要选择你的 Wi-Fi 所属的国家。如果你将使用以太网电缆，就无须这样做。确保你连接的网络与你的计算机所处的网络相同。

③ 勾选 **Set locale settings**（设置地区设置）复选框，并设置你的国家和时区。

④ 单击 **SAVE** 按钮。

（7）单击 **WRITE**（写入）按钮，将操作系统写到 SD 卡上。如果你使用的是 macOS，你将被要求输入登录密码，因为写磁盘镜像的底层工具需要访问权限。

操作系统将被写入 SD 卡，一旦完成，该卡将被操作系统弹出，你将得到通知。从你的计算机中取出 SD 卡，把它插入树莓派，给树莓派上电，等待大约 2 分钟，让它正常启动。

连接到树莓派

下一步是远程访问树莓派。你可以使用 ssh 来完成，它在 macOS、Linux 和 Windows 10 及以上版本的系统上都可以使用。

任务：连接到树莓派

远程访问树莓派。

（1）启动一个终端或命令提示符，并输入以下命令连接到树莓派。

```
ssh pi@raspberrypi.local
```

如果你在 Windows 上使用的是没有安装 ssh 的旧版本，你可以使用 OpenSSH。你可以在微软的安装 OpenSSH 的文档 [\[L1-14\]](#) 中找到安装说明。

（2）这样应该可以连接到你的树莓派，并按要求输入密码。

能够通过使用 **<hostname>.local** 来找到你网络上的计算机，是 Linux 和 Windows 最近增加的一项功能。如果你使用的是 Linux 或 Windows，当你遇到任何关于找不到主机名的错误时，你将需要安装额外的软件来启用 ZeroConf 网络（也被苹果称为 Bonjour）。

① 如果你使用的是 Linux，请使用以下命令安装 Avahi。

```
sudo apt-get install avahi-daemon
```

② 如果你使用的是 Windows，则启用 ZeroConf 的最简单方法是安装 Windows 版的 Bonjour Print Services [\[L1-15\]](#)。你也可以安装 Windows 版的 iTunes [\[L1-16\]](#)，以获得该工具的较新版本（该工具不能独立使用）。

③ 输入你在 Raspberry Pi Imager 高级选项中设置的密码。

 如果你不能使用 **raspberrypi.local** 连接，那么可以使用你的树莓派的 IP 地址。请参考树莓派的 IP 地址文档：[\[L1-17\]](#)，了解获取 IP 地址的多种方法。

在树莓派上配置软件

一旦连接到树莓派，你就需要确保操作系统是最新的，并安装与 Grove 硬件互动的各种库和工具。

任务：在树莓派上配置软件

配置已安装的树莓派软件并安装 Grove 库。

（1）在你的 ssh 会话中，运行以下命令来更新，然后重新启动树莓派。

```
sudo apt update && sudo apt full-upgrade --yes && sudo reboot
```

树莓派将被更新并重启。当树莓派被重启时，ssh 会话将结束，所以可以等待 30 秒左右，然后重新连接。

(2) 在重新连接的 **ssh** 会话中，运行以下命令为 Grove 硬件安装所有需要的库。

```
curl -sL https://gitee.com/seeed-projects/grove.py/raw/master/install.sh | sudo bash -s -
```

Python 的强大功能之一是能够安装 Pip 包 [L1-11]——这些是由其他人编写并发布到互联网上的代码包。你可以用一个命令将 Pip 包安装到你的计算机上，然后在你的代码中使用该包。这个 Grove 安装脚本将安装 Pip 包，你将使用 Python 与 Grove 硬件一起工作。

默认情况下，当你安装一个包时，它在你的计算机上各个应用程序中都是可用的，这可能会导致包的版本问题——比如一个应用程序依赖于一个包的版本，而当你为不同的应用程序安装一个新的版本时，这个包就会失效。为了解决这个问题，你可以使用一个 Python 虚拟环境 [L1-12]，基本上是在一个专门的文件夹中的 Python 副本，当你安装 Pip 包时，它们就会被安装到这个文件夹中。在使用你的树莓派时，你将不会使用虚拟环境。Grove 安装脚本会在全局范围内安装 Grove Python 软件包。所以，要使用虚拟环境，你需要建立一个虚拟环境，然后在该环境中手动重新安装 Grove 软件包。使用全局软件包会更简单，尤其是对于很多树莓派开发者会为每个项目新刷一张干净的 SD 卡的情况。

(3) 通过运行以下命令重新启动树莓派。

```
sudo reboot
```

当树莓派被重新启动时，**ssh** 会话将结束。没有必要重新连接。

配置 VS Code 进行远程访问

一旦树莓派配置好了，你就可以在你的计算机上使用 Visual Studio Code (VS Code) 来连接它——这是一个免费的开发者文本编辑器，你将使用它来编写 Python 设备代码。

任务：配置 VS Code 进行远程访问

安装所需的软件并远程连接到你的树莓派。

(1) 按照 VS Code 的文档：<https://code.visualstudio.com/>，在你的计算机上安装 VS Code。

(2) 按照 VS Code 远程开发使用 SSH 文档：<https://code.visualstudio.com/docs/remote/ssh> 中的说明，安装所需的组件。

(3) 按照 SSH 文档中的说明，将 VS Code 连接到树莓派上。

(4) 连接后，按照 SSH 文档中管理扩展的说明，将 Pylance 扩展 [L1-13] 远程安装到树莓派上。

Hello World

在开始使用一种新的编程语言或技术时，传统的做法是创建一个“Hello World”应用程序——一个输出类似“Hello World”文字的小应用程序，以显示所有工具都已正确配置。

树莓派的 Hello World 应用程序将确保你正确安装 Python 和 VS Code。

这个应用程序将被放在一个叫做 nightlight 的文件夹中，它将在本任务的后面部分用不同的代码重新使用，以建立 nightlight 应用程序。

任务：Hello World

创建 Hello World 应用程序。

(1) 启动 VS Code，可以直接在树莓派上启动，也可以在你的计算机上启动，并使用远程 SSH 扩展连接到树莓派上。

(2) 通过选择顶部菜单的“终端→新终端”命令，或按 **Ctrl+`** 组合键启动 VS Code 终端。它将打开到树莓派用户的主目录。

(3) 运行以下命令为你的代码创建一个目录，并在该目录下创建一个名为 **app.py** 的 Python 文件。

```
mkdir nightlight
cd nightlight
touch app.py
```

(4) 在 VS Code 中打开 nightlight 文件夹，选择“文件→打开”命令，选择“nightlight”文件夹，然后单击“OK”按钮，如图 1-23 所示。

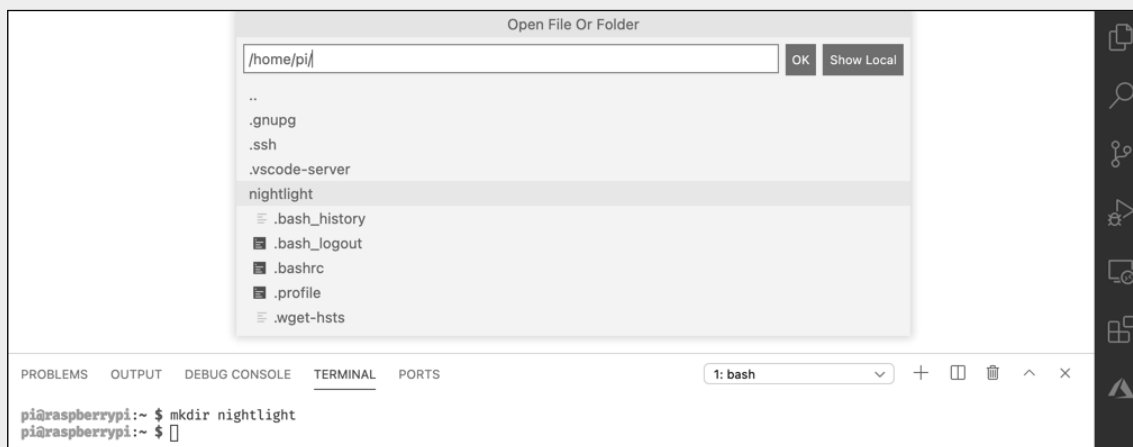


图 1-23 显示 nightlight 文件夹的 VS Code 打开对话框

(5) 从 VS Code 资源管理器中打开 **app.py** 文件，添加以下代码。

```
print('Hello World!')
```

print 函数将传递给它的东西打印到控制台。

(6) 从 VS Code 终端，运行下面的程序来运行你的 Python 应用程序。

```
python3 app.py
```

下面的输出将出现在终端上。

```
pi@raspberrypi:~/nightlight $ python3 app.py
Hello World!
```

你可以在 **code/pi** 文件夹 [\[L1-18\]](#) 中找到这段代码。

🎉 欢呼吧，你的“Hello World”程序很成功。

🐞 你需要明确地调用 Python3 来运行这段代码，以防你除了 Python3（最新版本）之外还安装了 Python2。如果你安装了 Python2，那么如果直接调用 Python，则将使用 Python2 而不是 Python3。

虚拟设备编程之 Hello World

如果你无法购买物联网设备以及传感器和执行器，可以用你的计算机来模拟物联网硬件。CounterFit 项目允许你在本地运行一个模拟物联网硬件的应用程序，如传感器和执行器，并通过本地 Python 代码访问传感器和执行器，该代码的编写方式与你在树莓派上使用物理硬件编写的代码相同。

设置

要使用 CounterFit，你需要在你的计算机上安装一些免费软件。

任务：安装所需的软件

(1) 安装 Python。请参考 Python 下载页面  [L1-19]，了解安装最新版本的说明。

(2) 安装 Visual Studio Code (VS Code)。这是你将用来在 Python 中编写虚拟设备代码的编辑器。关于安装 VS Code 的说明，请参考 VS Code 文档  [L1-20]。

(3) 安装 VS Code Pylance 扩展。这是 VS Code 的一个扩展， 如果你有喜欢的工具，你可以在这些课程中自由使用任何 Python IDE 或编辑器，但这些课程将基于 VS Code 给出教程。提供 Python 语言支持。请参考 Pylance 扩展文档  [L1-13] 中关于在 VS Code 中安装这个扩展的说明。

安装和配置 CounterFit 应用程序的说明将在作业指导中的适当时候给出，因为它是根据项目需要来安装的。

Hello world

在开始使用一种新的编程语言或技术时，传统的做法是创建一个“Hello World”应用程序——一个输出类似“Hello World”文字的小应用程序，以显示所有工具都已正确配置。

虚拟物联网硬件的 Hello World 应用程序将确保你正确安装 Python 和 Visual Studio Code。它还将连接到 CounterFit 的虚拟物联网传感器和执行器。它不会使用任何硬件，它只是通过连接以证明一切软件都在工作。

这个应用程序将被放在一个叫做 **nightlight** 的文件夹里，它将在本作业的后面部分在不同的代码中被重新使用，以建立 **nightlight** 应用程序。

配置一个 Python 虚拟环境

Python 的强大功能之一是安装 Pip 包的能力——这些是由其他人员编写并发布到互联网上的代码包。你可以用一个命令把 Pip 包安装到你的计算机上，然后在你的代码中使用该包。你将使用 Pip 来安装一个包与 CounterFit 对话。

默认情况下，当你安装一个包时，它在你的计算机上各个应用程序中都是可用的，这可能会导致包的版本问题——例如，一个应用程序依赖于一个包的某个版本，当你为一个不同的应用程序安装这个包的一个新的版本时，之前的应用程序会无法正常工作。为了解决这个问题，你可以使用一个 Python 虚拟环境  [L1-12]，本质上是在一个专门的文件夹中的 Python 副本，当你安装 Pip 包时，它们只被安装到该文件夹中。

 如果你使用的是树莓派，那么你并没有在该设备上设置一个虚拟环境来管理 Pip 包，而是使用了全局包，因为 Grove 包是由安装程序脚本全局安装。

任务：配置一个 Python 虚拟环境

配置一个 Python 虚拟环境，为 CounterFit 安装 Pip 包。

(1) 从你的终端或命令行，在你选择的位置运行下面的程序，创建并导航到一个新的目录。

```
mkdir nightlight
cd nightlight
```

(2) 现在运行下面的程序，在 `.venv` 文件夹中创建一个虚拟环境。

```
python3 -m venv .venv
```

🐘 你需要明确地调用 `python3` 来创建虚拟环境，以防你除了 Python3（最新版本）之外还安装了 Python2。如果你安装了 Python2，那么调用 `python` 时将使用 Python2 而不是 Python3。同时，我们建议你使用 Python3 以获得更好的体验。

(3) 激活虚拟环境。

- 在 Windows 上。

- 如果你使用命令提示符，或通过 Windows 终端使用命令提示符，运行：

```
.venv/Scripts/activate.bat
```

- 如果你使用的是 PowerShell，请运行：

```
.\.venv\Scripts\Activate.ps1
```

- 在 macOS 或 Linux 上。运行：

```
source ~/.venv/bin/activate
```

🐘 这些命令应该在你运行创建虚拟环境的命令的同一位置运行。你永远不需要导航到 `.venv` 文件夹中，你应该总是从你创建虚拟环境时所在的文件夹中运行激活命令和任何安装包或运行代码的命令。

⚠ 注意

如果用户在 PowerShell 终端尝试键入 `.\.venv\Scripts\Activate.ps1` 的时候出现如图 1-24 所示的错误信息，则可能是在计算机上启动 Windows PowerShell 时，执行策略很可能是 `Restricted`（默认设置），这个执行策略不允许任何脚本运行。



图 1-24 在 PowerShell 终端尝试键入 `.\.venv\Scripts\Activate.ps1` 的时候出现错误信息

AllSigned 和 **RemoteSigned** 执行策略可以防止 Windows PowerShell 运行没有数字签名的脚本。

本主题说明如何运行所选未签名脚本（即使在执行策略为 `RemoteSigned` 的情况下），还说明如何对脚本进行签名以便供你自己使用。

有关 Windows PowerShell 执行策略的详细信息，请参阅 [about_Execution_Policy](#) 文档 [L1-21]。

想了解计算机上的现用执行策略，打开 PowerShell，然后输入 `get-executionpolicy`，显示现有执行策略如下：

```
PS C:\Users\LeonFeng\nightlight> get-executionpolicy
Restricted
```

以管理员身份打开 PowerShell（在底部导航栏右击 PowerShell 图标，在弹出的菜单中选择“以管理员身份运行”），在管理员身份运行的 PowerShell 终端窗口中输入 `set-executionpolicy remotesigned`，如果有需要输入选择项，则输入 **Y**，如图 1-25 所示。



图 1-25 在 PowerShell 终端更改 RemoteSigned 执行策略

修改完毕后，再回到原来的 PowerShell 终端尝试输入 `.\.venv\Scripts\Activate.ps1`，可以看到虚拟环境被激活，如图 1-26 所示。

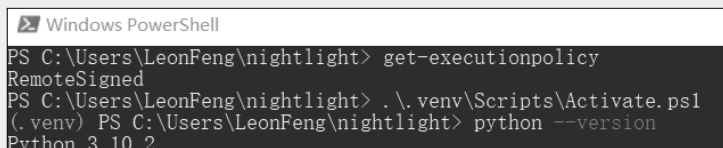


图 1-26 虚拟环境激活，出现 (.venv) 的提示

（4）一旦虚拟环境被激活，默认的 `python` 命令将运行用于创建虚拟环境的 Python 的版本。运行下面的命令来获取版本。

```
python --version
```

输出应该包含以下内容：

```
(.venv) → nightlight python --version
Python 3.9.1
```

👤 你的 Python 版本可能与笔者的一不一样——只要是 3.6 或更高版本就可以了。如果不是，则可以删除这个文件夹，安装一个较新版本的 Python，然后再试一次。

（5）运行下面的命令来安装 CounterFit 的 Pip 包。这些软件包包括主要的 CounterFit 应用程序以及用于 Grove 硬件的 shim（垫片）。这些垫片允许你编写代码，就像使用 Grove 生态系统中的物理传感器和执行器编程一样，但要连接到虚拟物联网设备。

```
pip install Counterfit
pip install counterfit-connection
pip install counterfit-shims-grove
```

这些 Pip 包将只在虚拟环境中安装，在虚拟环境之外将无法使用。

编写代码

一旦 Python 虚拟环境准备就绪，你就可以为“Hello World”应用程序编写代码了。

任务：编写代码

创建一个 Python 应用程序，向控制台打印“Hello World”。

(1) 从你的终端或命令行，在虚拟环境中运行以下程序，创建一个名为 app.py 的 Python 文件。

① 在 Windows 上运行。

```
type nul > app.py
```

② 在 macOS 或 Linux 上运行。

```
touch app.py
```

(2) 在 VS Code 中打开当前文件夹。

```
code .
```

🐛 如果你的终端在 macOS 上返回 command not found 信息，这意味着 VS Code 没有被添加到你的 PATH 中。你可以按照 VS Code 文档中从命令行启动部分的说明 [🔗 \[L1-22\]](#)，将 VS Code 添加到你的 PATH 中，然后运行命令。在 Windows 和 Linux 上，VS Code 被默认安装到你的 PATH 中。

(3) 当 VS Code 启动时，它将激活 Python 虚拟环境。选定的虚拟环境将出现在底部的状态栏中，如图 1-27 所示。

Python 3.9.1 64-bit ('.venv': venv)

图 1-27 VS Code 在右下角显示选定的虚拟环境

(4) 如果 VS Code 启动时 VS Code 终端已经在运行，那么它将不会激活虚拟环境。最简单的做法是使用 Kill the active terminal instance 按钮来杀终端进程，按钮位置如图 1-28 所示。

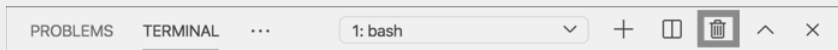


图 1-28 VS Code 的 Kill the active terminal instance 按钮

你可以知道终端是否激活了虚拟环境，因为虚拟环境的名称将是终端提示的前缀。例如，它可能是：

```
(.venv) → nightlight
```

如果提示符上没有 .venv 这个前缀，则说明终端中的虚拟环境没有被激活。

注意

请在虚拟环境中进行操作！

(5) 通过选择顶部菜单的“终端→新终端”选项，或按 **Ctrl+`** 组合键，启动一个新的 VS Code 终端。新的终端将加载虚拟环境，激活它的调用将出现在终端中。提示中还会有虚拟环境的名称 (.venv)。

```
→ nightlight source .venv/bin/activate
(.venv) → nightlight
```

在 VS 代码资源管理器中打开 `app.py` 文件，添加以下代码。

```
print('Hello World!')
```

`print` 函数将传递给它的东西打印到控制台。

在 VS Code 终端，通过运行下面的程序来运行你的 Python 应用程序。

```
python app.py
```

以下内容将出现在输出中。

```
(.venv) → nightlight python app.py  
Hello World!
```

🎉 恭喜，你的 'Hello World' 程序成功了！

连接“硬件”

作为第二个“Hello World”步骤，你将运行 CounterFit 应用程序并将你的代码连接到它。这个过程是虚拟的，相当于把一些物联网硬件插到开发套件上。

任务：连接“硬件”

(1) 在 VS Code 终端，用以下命令启动 CounterFit 应用程序。

```
counterfit
```

(2) 将该应用打开：在你的网络浏览器中输入以下网址：`localhost:5000`。该应用将开始运行，并在你的网络浏览器中显示如图 1-29 所示的内容。

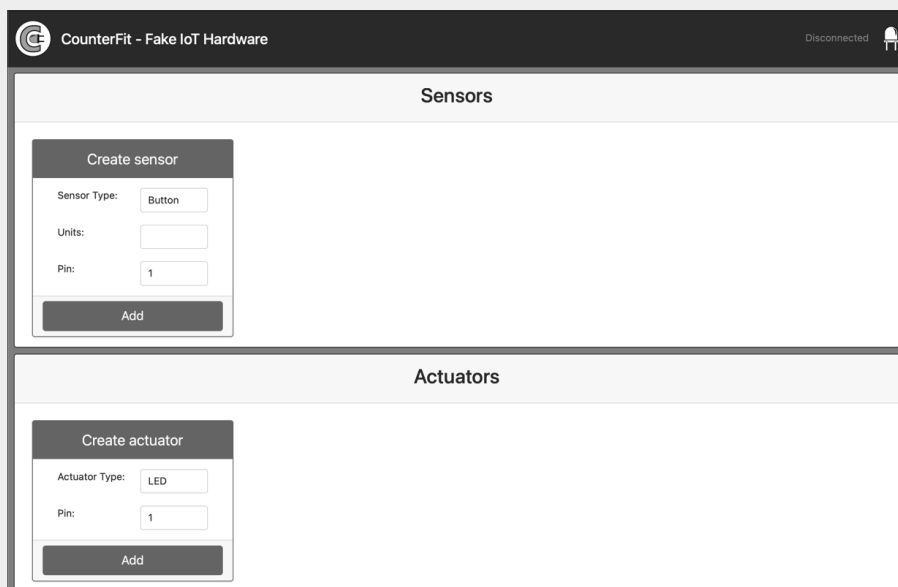


图 1-29 在浏览器中运行的 CounterFit 应用程序

⚠ 注意

如果用户在 VS Code 终端尝试键入 CounterFit 的时候出现以下错误，可以通过下面的操作进行处理。

```
(.venv) (base) mouseart@LeondeMacBook-Pro nightlight % counterfit
Traceback (most recent call last):
  File "/Users/mouseart/nightlight/.venv/bin/counterfit", line 5, in <module>
    from CounterFit.counterfit import main
  File "/Users/mouseart/nightlight/.venv/lib/python3.9/site-packages/CounterFit/counterfit.py",
line 13, in <module>
    from flask import Flask, request, render_template
  File "/Users/mouseart/nightlight/.venv/lib/python3.9/site-packages/flask/__init__.py", line
19, in <module>
    from . import json
  File "/Users/mouseart/nightlight/.venv/lib/python3.9/site-packages/flask/json/__init__.py",
line 15, in <module>
    from itsdangerous import json as _json
ImportError: cannot import name 'json' from 'itsdangerous' (/Users/mouseart/nightlight/.venv/
lib/python3.9/site-packages/itsdangerous/__init__.py)
```

● Windows 用户。

- ° 首先打开 Powershell，键入：`pip3 uninstall Flask`。
- ° 然后安装 Flask==1.14，请忽视兼容性警告，安装命令：`pip3 install Flask==1.14`。
- ° 最后再次在 venv 下运行 counterfit 命令。

● macOS 用户。

- ° 首先打开终端，键入：`pip3 uninstall Flask pip3 uninstall markupsafe`。
- ° 然后安装 Flask==1.1.4 和 markupsafe==2.0.1，请忽视兼容性警告：`pip3 install Flask==1.1.4`。
- ° `pip3 install markupsafe==2.0.1`。
- ° 最后再次在 venv 下运行 `counterfit` 命令。

⚠ 注意

如果出现 CounterFit 默认端口（5000）被占用，如下面的提示：

```
(.venv) (base) mouseart@LeondeMacBook-Pro nightlight % counterfit
CounterFit - virtual IoT hardware running on port 5000
Traceback (most recent call last):
  File "/Users/mouseart/nightlight/.venv/bin/counterfit", line 8, in <module>
    sys.exit(main())
  File "/Users/mouseart/nightlight/.venv/lib/python3.9/site-packages/CounterFit/counterfit.py",
line 456, in main
    socketio.run(app, port=args.port, host='0.0.0.0')
  File "/Users/mouseart/nightlight/.venv/lib/python3.9/site-packages/flask_socketio/__init__.
py", line 621, in run
    run_server()
  File "/Users/mouseart/nightlight/.venv/lib/python3.9/site-packages/flask_socketio/__init__.
py", line 599, in run_server
    eventlet_socket = eventlet.listen(addresses[0][4],
  File "/Users/mouseart/nightlight/.venv/lib/python3.9/site-packages/eventlet/convenience.py",
line 78, in listen
    sock.bind(addr)
```

```
OSError: [Errno 48] Address already in use
(.venv) (base) mouseart@LeondeMacBook-Pro nightlight %
```

🐞 你可以通过更新代码中的端口来改变（不超过 25565），并使用 `counterfit --port <port_number>` 运行 CounterFit，例如：`counterfit --port 8000`。终端显示如下：

```
(.venv) (base) mouseart@LeondeMacBook-Pro nightlight % counterFit --port 8000
CounterFit - virtual IoT hardware running on port 8000
Sensor units called: {'type': 'Button'}
```

（3）在 `app.py` 的顶部添加以下代码。

```
from counterfit_connection import CounterFitConnection
CounterFitConnection.init('127.0.0.1', 5000)
```

这段代码从 `counterfit_connection` 模块中导入 `CounterFitConnection` 类，该模块来自于你之前安装的 `counterfit-connectionpip` 包。然后，它初始化了与运行在 `127.0.0.1` 上的 CounterFit 应用程序的连接，这是一个你可以一直用来访问你的本地计算机的 IP 地址（通常称为 localhost），端口为 5000。

（4）你需要通过选择创建一个新的集成终端按钮来启动一个新的 VS Code 终端，按钮位置如图 1-30 所示。这是因为 CounterFit 应用程序正在当前终端中运行。



图 1-30 VS Code 创建一个新的集成终端按钮

（5）在这个新终端中，像前述一样运行 `app.py` 文件。刷新浏览器，CounterFit 的状态将变为 Connected（连接），右侧的 LED 灯将为亮起状态，如图 1-31 所示。

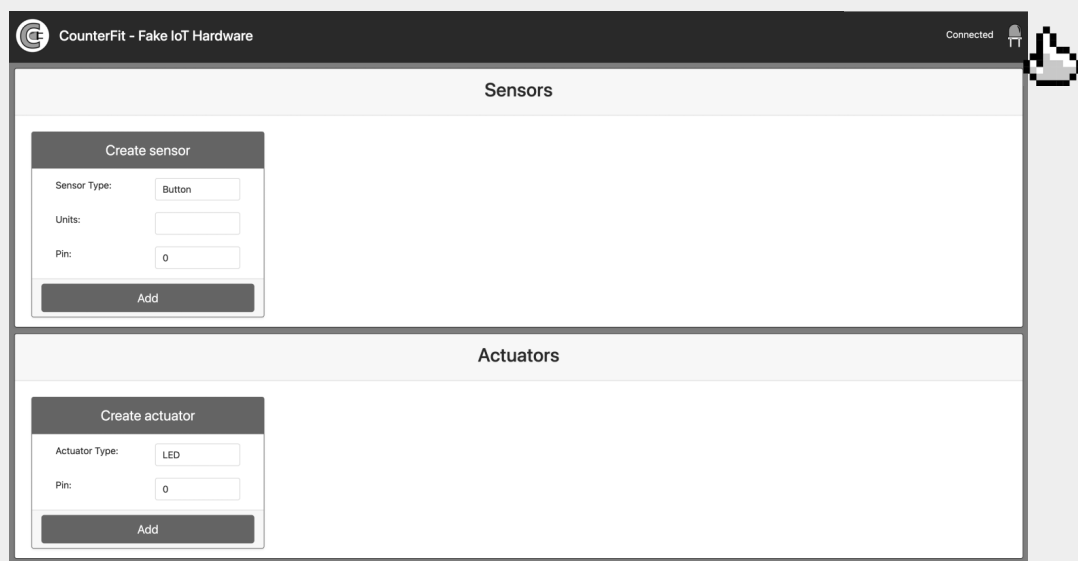


图 1-31 CounterFit 显示为已连接

你可以在 `code/virtual-device` 文件夹  [L1-23] 中找到这个代码。

 祝贺你，你与虚拟硬件的连接成功了！

1.4 物联网的应用

物联网涵盖了大量的使用案例，跨越了以下几个大的群体。

- 消费级物联网。
- 商业物联网。
- 工业物联网。
- 基础设施物联网。

做点研究

对于下面描述的每个领域，请找到一个本文中未给出的具体例子。

1.4.1 消费级物联网

消费级物联网（Consumer IoT）指的是消费者购买并在家庭环境中使用的物联网设备。其中一些设备是非常有用的，如智能音箱、智能空调系统和扫地机器人。但是其他一些设备的实用性值得怀疑，如声控水龙头，这意味着一旦打开你将无法关闭它们，因为语音控制无法在流水声中听到你的指令。

消费级物联网设备正在促进人们在其周围环境中实现更多的功能，特别是 10 亿残疾人。扫地机器人可以为行动不便而无法自己吸尘的人们提供干净的地面，语音控制的烤箱可以让视力或运动控制能力有限的人只用声音即可实现加热，健康监测仪可以让患者自己监测慢性病，更规律并且更详细地更新他们的病情数据。这些设备正变得无处不在，甚至年幼的孩子也在使用它们作为日常生活的一部分，例如，在新型冠状病毒感染大流行期间，进行虚拟学校教育的学生在智能家居设备上设置计时器来跟踪他们的学校作业，或者用定时闹钟提醒他们即将到来的班级会议。

 你的身上或家里有哪些消费级物联网设备？

1.4.2 商业物联网

商业物联网（Commercial IoT）包括物联网在工作场所的使用。在办公室环境中，可能有人体感应传感器和运动感应器来管理照明和暖气，在不需要的时候可以进行关灯和停止供暖，减少成本和碳排放。在工厂里，物联网设备可以监测安全隐患，如工人不戴安全帽或噪声达到危险水平。在零售业，物联网设备可以测量冷库的温度，如果冰箱或冰柜超出规定的温度范围，就会提醒店主，或者它们可以监测货架上的物品，指导员工补货。运输业正越来越多地依赖物联网设备来监测车辆的位置，跟踪按道路使用量收费的车辆的公路里程，跟踪驾驶员工作时间和休息时间的遵守情况，或在车辆接近仓库时通知工作人员准备装货或卸货。

 你的学校或工作场所有哪些商业物联网设备？

1.4.3 工业物联网

工业物联网（Industrial IoT）也称 IIoT，是指使用物联网设备控制和管理大规模的机械。这涵盖了从工厂到数字农业的诸多案例。

工厂以许多不同的方式使用物联网设备。机器可以用多个传感器进行监测，以跟踪温度、振动和转速等项目。然后这些数据可以被监测，以便设备数据在超出某些公差时被停止。例如，机器会因为运行过热而被关闭。这些数据也可以被收集起来，并随着时间的推移进行分析，以便进行预测性维护，其中人工智能模型将查看导致故障的数据，并利用这些数据在发生故障之前预测其他故障。

 还有哪些物联网设备可以帮助农民？

如果地球要养活不断增长的人口，那么数字农业是很重要的，特别是对于 5 亿家庭中靠自给自足的农业生存的 20 亿人口。数字化农业的范围可以从几个几元钱的传感器到大规模的商业设备。农民可以从监测温度开始，利用生长度日来预测作物何时可以收获。他们可以将土壤湿度监测与自动浇水系统连接起来，给植物提供所需的水，且不会过多，以确保他们的作物在不会干枯的同时也不会浪费水。农民甚至更进一步，使用无人机、卫星数据和人工智能来监测大片农田中的作物生长、疾病和土壤质量。

1.4.4 基础设施物联网

基础设施物联网（Infrastructure IoT）用于监测和控制人们每天使用的本地和全球基础设施。

智慧城市是指在城市范围内使用物联网设备来收集城市的数据，并利用这些数据来改善城市的运行方式。这些城市通常由当地政府、学术界和当地企业合作经营，跟踪和管理从交通到停车以及污染等各种领域。例如，在丹麦的哥本哈根，空气污染对当地居民很重要，所以物联网设备被广泛应用于测量，数据被用于提供出空气质量最好的自行车和慢跑路线信息。

智能电网通过收集个人家庭层面的使用数据，可以更好地分析电力需求。这些数据可以指导国家层面的决策，包括在哪里建设新的发电站，也可以指导个人层面的决策，让用户了解他们正在使用多少电力，何时使用，甚至建议如何降低成本，如在夜间为电动汽车充电等。

1.5 探究你身边的物联网设备

- 智能音箱。
- 冰箱、洗碗机、烤箱和微波炉。
- 太阳能电池板的电力监测器。
- 智能插头。
- 视频门铃和安防摄像头。
- 带有多个智能房间传感器的智能恒温器。
- 车库门开启器。
- 家庭娱乐系统和语音控制的电视。
- 灯光。
- 健身和健康追踪器。

 你会用什么物联网设备来记录你生活的方方面面？

所有这些类型的设备都有传感器和 / 或执行器，并且有与互联网交流的功能。我可以从手机上知道车库门是否打开，并要求智能音箱关闭它。我甚至可以把它设置成一个定时器，这样如果它在晚上还开着，它就会自动关闭。当门铃响起时，我可以从手机上看到谁站在那里，无论我在哪里，都可以通过门铃内置的扬声器和麦克风与他们交谈。我可以监测我的血糖、心率和睡眠模式，在数据中寻找规律，以改善我的健康。我可以通过云计算控制我的灯光。但是当家里断网时，就无计可施了。



课后练习

挑战

尽可能多地列举出你家中、学校或工作场所中的物联网设备——这可能比你想象中的要更多一些！

复习和自学

阅读有关消费级物联网项目的优劣信息。检索新闻门户网站上与消费级物联网有关的负面信息，如隐私问题、硬件问题或连接缺失造成的问题。一些例子如下。

- 查看推特账户 Internet of Sh*t [\[L1-24\]](#)（警告：可能包含脏话），了解一些消费者物联网失败的好例子。
- c|net – 我的苹果手表救了我的命：5 人分享他们的故事 [\[L1-25\]](#)。
- c|net – ADT 技术员承认多年来偷窥客户的摄像头画面（触发警告——非自愿的偷窥） [\[L1-26\]](#)。

课后测验

（1）IoT 设备需要一直连接着网络。这个说法（ ）。

- A. 正确
- B. 错误

（2）IoT 设备总是安全的。这个说法（ ）。

- A. 正确
- B. 错误

（3）人工智能（AI）可以在低功耗的 IoT 设备中运行。这个说法（ ）。

- A. 正确
- B. 错误

作业

调查一个物联网项目。

说明

有许多大大小小的物联网项目正在全球范围内展开，涉及智能农场、智能城市，存在于医疗监控、交通以及公共场所使用的方方面面。

在网上搜索你感兴趣的项目的细节，最好是你身边的项目。解释这个项目的优点和缺点。例如，它可以带来什么好处，它会造成什么样的问题，以及如何考虑到隐私。

评分标准

标准	优秀	合格	需要改进
解释优点和缺点	对项目的优点和缺点作了明确的解释	对优点和缺点作了简要的解释	没有解释优点或缺点

第 2 课

深入了解物联网



课前准备

本课是微软 Reactor 的 Hello IoT 系列课程的一部分。课程通过 2 个视频（英文）对物联网做了深入探讨。



观看视频：更深入地了解物联网  [L2-1]



观看视频：自学时间  [L2-2]

课前测验

(1) IoT（物联网）中的“T”代表（ ）。

- A. 晶体管（Transistors）
- B. 万物（Things）
- C. 火鸡（Turkeys）

(2) IoT 设备使用（ ）获取其周围世界的信息。

- A. 传感器
- B. 执行器
- C. 信鸽

(3) IoT 设备通常比台式计算机或笔记本更耗电。这个说法（ ）。

- A. 正确
- B. 错误

简介

本课深入探讨了上一课中所涉及的一些概念。

在本课中，我们的学习内容将涵盖：

- 2.1 物联网应用的组成部分
- 2.2 深入了解微控制器
- 2.3 深入了解单板计算机

2.1 物联网应用的组成部分

物联网应用的两个组成部分是“互联网”和“物”。让我们更详细地了解一下这两个组成部分。

2.1.1 物

物联网的“物”是指能够与物理世界互动的设备，如图 2-1 所示的树莓派。这些设备通常是小型、低价的计算机，它们以低速运行，功耗很低。例如，简单的微控制器，有几千字节的内存（相对于 PC 的几吉字节的内存），运行速度只有几百兆赫（相对于 PC 的几千兆赫），

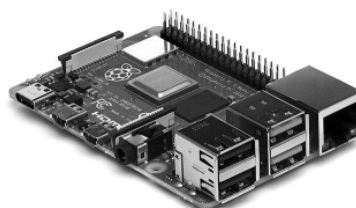


图 2-1 树莓派 4

但功耗却小得多，可以用电池运行几周、几个月甚至几年。

这些设备与物理世界互动，通过传感器从其周围环境收集数据，或者通过控制输出或执行器来进行物理控制。这方面的典型例子是自动调温器，这种设备有一个温度传感器、一个设置所需温度的设备，如表盘（或触摸屏），以及将表盘和温度传感器与加热或冷却系统相连接，当检测到的温度超出所需范围时便会启动。若温度传感器检测到房间太冷，会通过一个执行器将加热功能打开，如图 2-2 所示。

做点研究

想一想你身边有哪些从传感器中读取数据，并利用这些数据来做出控制的物联网系统。例如，烤箱上的恒温器。除此之外，你能找到更多吗？

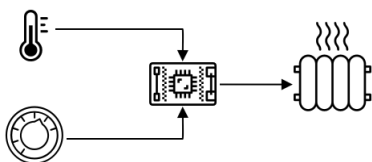


图 2-3 温度传感器和表盘作为物联网设备的输入，以及控制加热器作为输出

有多种设备可以作为物联网设备，从感知事件的专用硬件，到通用设备，甚至是你的智能手机！一部智能手机可以使用传感器来检测房间的温度。智能手机可以使用传感器来检测周围的世界，并使用执行器与世界进行互动——例如，使用 GPS 传感器来检测你的位置，并使用扬声器向你提供前往目的地的导航指示。

2.1.2 互联网

物联网应用的互联网方面包括物联网设备可以连接的用于发送和接收数据的应用程序，以及其他可以处理来自物联网设备数据的应用程序，并帮助决定向物联网设备的执行器发送什么请求。

一种典型的设置是让物联网设备连接到某种云服务，这种云服务处理包括安全、接收来自物联网设备的消息以及将消息发送回设备等问题。然后，该云服务将连接到其他应用程序，这些应用程序可以处理或存储传感器数据，或者将传感器数据与其他系统的数据一起使用，以做出决策。

设备也不总是通过 Wi-Fi 或有线网络连接直接连接到互联网本身。一些设备使用多跳网络（Mesh Network），通过蓝牙等技术相互交流，再通过一个有互联网连接的枢纽设备进行连接。

以智能自动调温器为例，它将使用家庭 Wi-Fi 连接到在云中运行的云服务。它将把温度数据发送到该云服务，并在云服务中将其写入某种数据库，以便能够让房主使用手机应用程序查询当前温度和历史温度。云中的另一项服务是可以知道房主想要什么温度，并通过云服务向物联网设备发送消息，控制供暖或制冷系统打开或关闭，如图 2-3 所示。

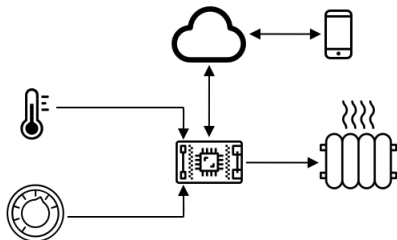


图 2-2 云服务引入物联网系统

一个更智能的版本可以在云中使用人工智能，将其他传感器的数据连接到其他物联网设备上，如检测哪些房间正在使用人体感应传感器、天气甚至你的日历等数据，来决定如何以智能的方式设置温度，如图 2-4 所示。例如，如果它从你的日历中读到你在度假，它就可以关闭你的暖气，或者根据你使用的房间逐个关闭暖气，随着时间的推移，它将通过学习数据变得越来越准确。

做点研究

还有哪些数据可以使互联网连接的自动调温器更加智能？

2.1.3 边缘物联网

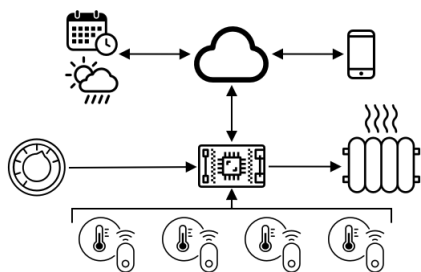


图 2-4 多个温度传感器和一个表盘作为物联网设备的输入，物联网设备与云端进行双向通信，云端又与手机、日历和天气服务进行双向通信，控制加热器作为物联网设备的输出

虽然 IoT 中的 I 代表互联网，但这些设备不一定要时刻连接到互联网。在某些情况下，设备可以连接到“边缘(Edge)”设备——在你的本地网络上运行的网关设备，这意味着你可以处理数据而不需要通过互联网连接。当你有大量数据或互联网连接缓慢时，这种模式可能会更快，它允许你在无法连接互联网的地方离线运行，如在船上或在应对人道主义危机的灾区，并允许你保持数据的隐私。一些设备将包含使用云工具创建的处理代码，并在本地运行这些代码来收集和响应数据，而不使用互联网连接来做出决策。

这方面的一个例子是智能家居设备，如苹果 HomePod、亚马逊的 Alexa、谷歌的 Home 或小米的米家等，它将使用在云端训练的人工智能模型聆听你的声音，但在设备上本地运行。这些设备会在说某个单词或短语时被“唤醒”，然后才通过互联网发送你的语音进行处理。该设备将在适当的时候停止发送语音，如当它检测到你的语音暂停时。你在用唤醒词唤醒设备之前所说的一切，以及你在设备停止监听之后所说的一切，都不会通过互联网发送给设备提供商，因为这些都是隐私。

做点研究

想一想其他的场景，在这些场景中，隐私是很重要的，所以数据的处理最好在边缘而不是在云端进行。作为一个提示——想想物联网设备上的摄像头或其他成像设备。

2.1.4 物联网安全

对于任何互联网连接，安全是一个重要的考虑因素。有一个老梗：

“IoT（物联网）中的 S 代表安全（Security）”——因为 IoT 中没有“S”，所以意味着它不安全。

物联网设备要连接到云服务，因此只有该云服务才是安全的——如果你的云服务允许任何设备连接，那么恶意数据就能够被发送，或者可能出现病毒攻击。这可能会产生非常现实的后果，因为物联网设备与其他设备进行互动和控制。例如，Stuxnet 蠕虫病毒操纵离心机的阀门来破坏它们（这个病毒对伊朗的核计划造成重大危害）。黑客们还利用糟糕的安全性来访问婴儿监视器和其他家庭监控设备 [L2-3]。

有时，物联网设备和边缘设备在一个与互联网完全物理隔离的网络上运行，以保持数据的私密性和安全性，这被称为 air-gapping（气隙）。

2.2 深入了解微控制器

在上一课中，我们介绍了微控制器。现在让我们更深入地了解它们。

2.2.1 中央处理器 CPU

CPU 是单片机的“大脑”。它是运行代码的处理器，可以向任何连接的设备发送数据和接收数据。CPU 可以包含一个或多个内核——基本上是一个或多个 CPU，它们可以一起工作来运行你的代码。

CPU 依靠时钟信号每秒跳动数百万至数十亿次不等。每个滴答声

CPU 速度通常以兆赫（MHz，1MHz 是 100 万赫兹）或千兆赫（GHz，1GHz 是 10 亿赫兹）为单位。

或循环都会同步 CPU 可以执行的操作。在每一个时钟周期中，CPU 都可以执行来自程序的指令，如从外部设备检索数据或执行数学计算。此常规周期允许在处理下一条指令之前完成所有操作。

时钟周期越快，每秒可以处理的指令就越多，因此 CPU 的速度也越快。CPU 的速度以赫兹 (Hz) 为单位，这是一个标准单位，其中 1Hz 表示每秒一个时钟周期。

微控制器的时钟速度比台式计算机或笔记本，甚至大多数智能手机都要低得多。例如，Wio Terminal 的 CPU 运行速度为 120 兆赫 (MHz)，或者说每秒 1.2 亿次。

每个时钟周期都会耗费电能并产生热量。节拍越快，消耗的电力越多，产生的热量也越多。PC 有散热器和风扇来清除热量，如果没有这些，它们会在几秒钟内过热并关闭。微控制器通常没有这些，因为它们的运行温度低得多，因此速度也慢得多。PC 靠主电源或大电池运行几个小时，而微控制器靠小电池可以运行几天、几个月，甚至几年。微控制器也可以有以不同速度运行的内核，当对 CPU 的需求较低时，切换到较慢的低功率内核，以减小功耗。

一些 PC 正采用同样的快速高功率内核和较慢的低功率内核的组合，切换以节省电池。例如，最新的苹果笔记本中的 M1 芯片（见图 2-6）可以在 4 个性能核心和 4 个效率核心之间切换，以根据运行的任务优化电池寿命或速度。

任务：研究 Wio Terminal

研究 Wio Terminal。

如果你在这些课程中使用 Wio Terminal，那么可以试着找到 CPU。找到 Wio Terminal 的 Wiki 页面 [🔗 \[L2-5\]](#) 中“硬件概述”部分的内部图片，并尝试通过背面的透明塑料窗找到 CPU。

2.2.2 存储器

微控制器通常有两种类型的存储器——程序存储器和随机存取存储器（RAM）。程序存储器是非易失

🔗 CPU 使用“获取—解码—执行”指令周期执行程序（见图 2-5）。每一个时钟节拍，CPU 将从内存中获取下一条指令，对其进行解码，然后执行它，例如使用算术逻辑单元（ALU）来添加 2 个数字。有些执行需要多个时钟节拍，所以下一个周期将在指令完成后的下一个节拍运行。

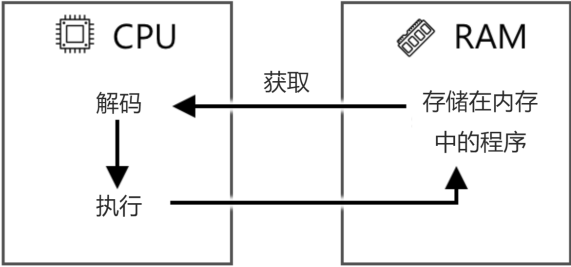


图 2-5 CPU 的“获取—解码—执行”指令周期执行程序

🔗 做点研究

普通 PC 或 Mac 的 CPU 有多个核心，以多个千兆赫 (GHz) 的速度运行，这意味着时钟每秒跳动数十亿次。研究一下你计算机的时钟速度，比较一下它比 Wio Terminal 快多少。



图 2-6 M1 是第一款使用尖端 5 纳米工艺技术制造的 PC 芯片，其中包含惊人的 160 亿个晶体管

🔗 做点研究

做一点研究。阅读百度百科 CPU（中央处理器）的文章 [🔗 \[L2-4\]](#)。

性的，这意味着当设备没有电源时，写入它的任何内容都会保留。它是存储程序代码的存储器。

RAM 是程序运行时使用的内存，包含你的程序分配的变量和从外围设备收集的数据。RAM 是易失性的，当电源中断时，内容就会丢失，有效地重置了你的程序。

与 CPU 一样，微控制器的内存比 PC 小几个数量级。一台典型的 PC 可能有 8 GB 的 RAM，或者说 8 000 000 000 字节的 RAM，每个字节的空间足以存储一个字母或 0 ~ 255 的数字。一个微控制器只有几千字节 (KB) 的 RAM，一个千字节是 1 024 字节。上面提到的 Wio Terminal 有 192KB 的 RAM，约 192 000 字节。

图 2-7 所示显示了 192KB 和 8GB 之间的相对大小差异——中间的小点代表 192KB。

微控制器程序存储空间也比 PC 小。一台典型的 PC 可能有 500GB 的硬盘用于程序存储，而一个微控制器可能只有几千字节或几兆字节 (MB) 的存储空间 (1MB 是 1 000KB，或 1 000 000 字节)。Wio Terminal 有 4MB 的程序存储空间。

- 程序存储器储存你的代码，在没有电源时保持不变。
- RAM 用于运行你的程序，在没有电源时会被重置。



图 2-7 192KB 和 8GB 之间的比较 (超过 40 000 倍的大小)

2.2.3 输入 / 输出

微控制器需要输入和输出 (I/O) 连接，以便从传感器读取数据，并向执行器发送控制信号。它们通常包含一些通用的输入 / 输出 (GPIO) 引脚。这些引脚可以在软件中配置为输入 (即接收信号) 或输出 (即发送信号)。

- ← 输入引脚用于从传感器读取数值。
- 输出引脚向执行器发送指令。

任务：研究 Wio Terminal

如果你在这些课程中使用 Wio Terminal，则可以找到 Wio Terminal Wiki 页面 [\[L2-6\]](#) 的引脚图部分，如图 2-8 所示，了解那些引脚是什么。Wio Terminal 有一张贴纸，你可以把它贴在背面，上面有引脚编号，如果你还没有，可以现在就把它加上去。

2.2.4 物理尺寸

微控制器的尺寸通常较小，最小的飞思卡尔 Kinetis KL03 微控制器 [\[L2-7\]](#) 小到可以放在高

输入引脚用于从传感器读取数值。输出引脚向执行器发送指令。你将在随后的课程中了解更多信息。

做点研究

你的 PC 有多少内存和存储空间？这与微控制器相比有什么不同？

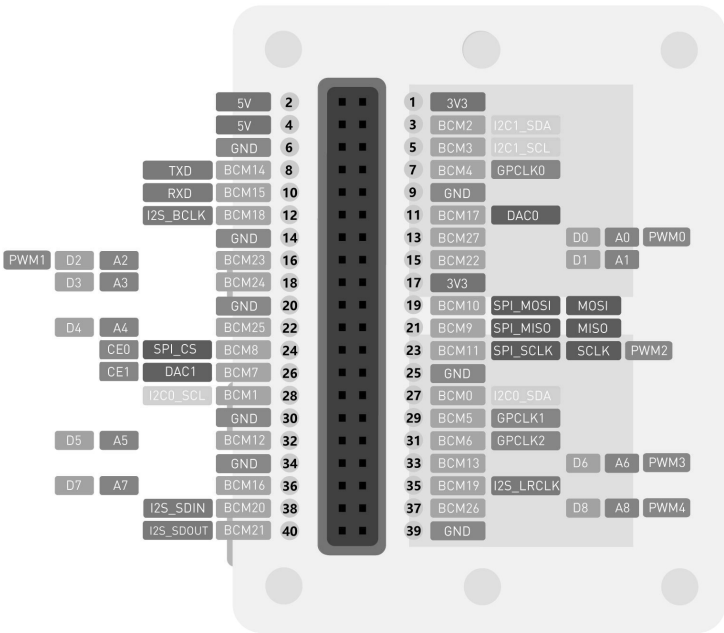


图 2-8 Wio Terminal 的引脚图

尔夫球的凹陷处,如图2-9所示。而 PC 中仅仅 CPU 就可以达到 40mm × 40mm,这还不包括为确保 CPU 能够运行超过几秒钟而不会过热所需的散热器和风扇,其尺寸大大超过了一个完整的微控制器。包含微控制器、外壳、屏幕和一系列连接和组件的 Wio Terminal 开发套件并不比一个裸露的英特尔 i9 CPU 多多少,而且大大小于带有散热片和风扇的 CPU。

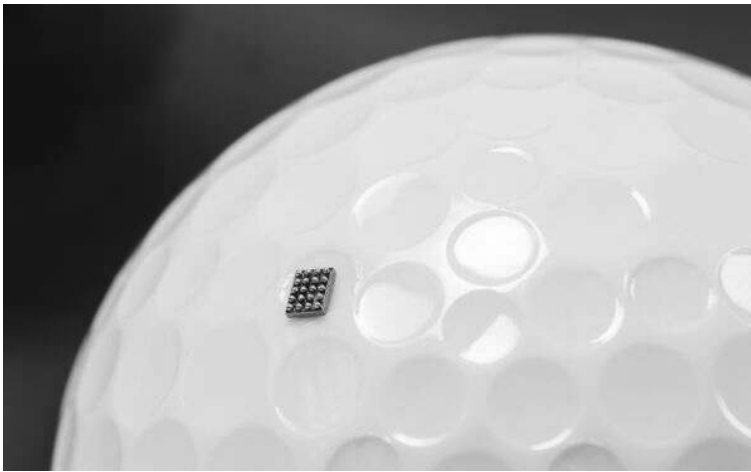


图 2-9 飞思卡尔 Kinetis KL03 在高尔夫球凹陷处

2.2.5 框架和操作系统

由于其低速和内存大小,微控制器并不运行桌面意义上的操作系统 (OS)。使计算机运行的操作系统 (Windows、Linux 或 macOS) 需要大量的内存和处理能力来运行对于微控制器来说完全不必要的任务。记住,微控制器通常被编程为执行一个或多个非常具体的任务,不像 PC 这样的通用计算机需要支持用户界面、播放音乐或电影、提供编写文档或代码的工具,玩游戏或浏览互联网。

器件	尺寸
飞思卡尔 Kinetis KL03	1.6 mm × 2 mm × 0.56 mm
Wio Terminal	72 mm × 57 mm × 12 mm
英特尔 i9 CPU, 散热器和风扇	136 mm × 145 mm × 103 mm

要为没有操作系统的微控制器编程,你确实需要一些工具,让你以微控制器可以运行的方式构建你的代码,使用可以与任何外设对话的 API。每个微控制器都是不同的,所以制造商通常支持标准的框架,允许你按照标准的“配方”来构建你的代码,让它在任何支持该框架的微控制器上运行。

你可以使用操作系统对微控制器进行编程——通常被称为实时操作系统 (RTOS),因为这些操作系统被设计为实时处理与外设之间的数据发送。这些操作系统非常轻量级,并提供以下功能。

- 多线程,允许你的代码同时运行一个以上的代码块,可以在多个核心上运行,也可以在一个核心上轮流运行。
- 网络,允许在互联网上安全地进行通信。
- 图形用户界面组件,用于在有屏幕的设备上构建用户界面 (UI)。

做点研究

了解一些不同的实时操作系统。例如, Azure RTOS [L2-8], FreeRTOS [L2-9], Zephyr [L2-10] 等。

2.2.6 Arduino

Arduino [L2-11] 可能是最流行的微控制器框架,特别是在学生、业余爱好者和创客中,标志如图 2-10 所示。Arduino 是一个开源的电子平台,结合了软件和硬件。你可以从 Arduino 本身或其他制造商处购买 Arduino 兼容板,然后使用 Arduino 框架进行编码。

Arduino 板是用 C 或 C++ 编码的。使用 C/C++ 允许你的代码被编译得非常小,而且运行速度快,这是在微控制器这样的受限设备上需要的特性。Arduino 应用程序的核心被称为 sketch (代码),是具



图 2-10 Arduino 的标志

有两个函数的 C/C++ 代码——`setup`（设置）和 `loop`（循环）。当电路板启动时，Arduino 框架代码将运行一次设置函数，然后它将一次又一次地运行循环函数，连续运行直到电源关闭，如图 2-11 所示。

你将在 `setup` 函数中编写你的设置代码，如连接到 Wi-Fi 和云服务，或初始化输入和输出的引脚。然后，你的循环代码将包含处理代码，如从传感器中读取数值并将其发送到云端。通常会在每个循环中包含一个延迟，例如，如果你只想每 10 秒发送一次传感器数据，则在循环结束时添加一个 10 秒的延迟，这样微控制器就可以休眠，节省电量，然后在 10 秒后需要时再次运行循环。

Arduino 提供了与微控制器和 I/O 引脚交互的标准库，在不同的微控制器上运行不同的实现。例如，`delay`（延迟）函数将暂停程序一段时间，`digitalRead`（数字读取）函数将从给定的引脚读取一个 `HIGH`（高）或 `LOW`（低）的值，无论代码在哪个板上运行。这些标准库意味着为一个板子编写的 Arduino 代码可以为任何其他 Arduino 板子重新编译并运行，但前提是引脚相同，板子支持同样的功能。

Arduino 有一个庞大的基于第三方 Arduino 库的生态系统，允许你为你的 Arduino 项目添加额外的功能，如使用传感器和执行器或连接到物联网云服务。

任务：研究 Wio Terminal

如果你在这些课程中使用 Wio Terminal，则重新阅读你在上一课中写的代码。找到 `setup` 和 `loop` 的功能。监测反复调用的循环函数的串行输出。试着在 `setup` 函数中添加代码以输出(字符串)到串口，并观察到每次重启时只调用一次该代码。试着用侧面的电源开关重启你的设备，以证明每次设备重启时都会调用这部分内容。

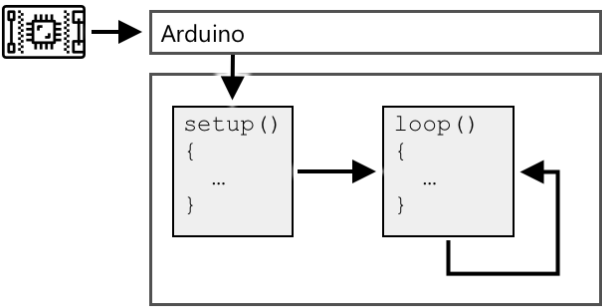


图 2-11 一个 Arduino 代码先运行 `setup`（设置）部分的程序，然后再重复运行 `loop`（循环）部分的程序

做点研究

这种程序结构被称为事件循环或消息循环。许多应用程序都内在地使用它，它是大多数运行在 Windows、macOS 或 Linux 等操作系统上的桌面应用程序的标准。循环监听来自用户界面组件（如按钮）或设备（如键盘）的消息，并对其做出响应。你可以在这篇文章 [\[L2-12\]](#) 中阅读更多关于事件循环的内容。

2.3 深入了解单板计算机

在上一课中，我们介绍了单板计算机。现在让我们更深入地了解它们。

2.3.1 树莓派

树莓派基金会 [\[L2-13\]](#) 是英国的一个慈善机构，成立于 2009 年，旨在促进计算机科学的学习，特别是在学校层面，图标如图 2-12 所示。作为这一使命的一部分，他们开发了一种称为 Raspberry Pi（树莓派）的单板计算机。树莓派目前有三个版本，即全尺寸版本、较小的 Pi Zero，以及可以内置到你的最终物联网设备中的计算模块 CM4。另

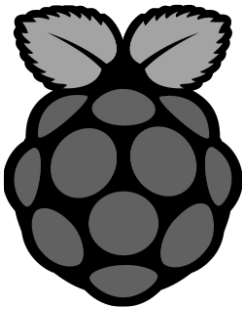


图 2-12 树莓派的标志

外还有一个内置了树莓派 4B 的看着像个键盘的一体机设备 Pi 400，如图 2-13 所示。

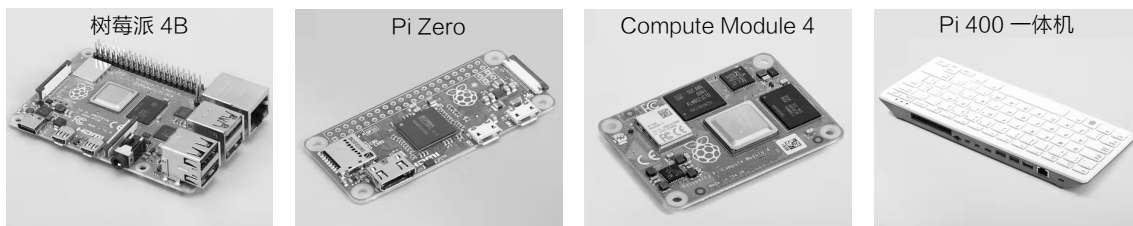


图 2-13 树莓派的三个版本和一体机设备

树莓派 4

全尺寸树莓派的最新迭代版本是树莓派 4B。它有一个 1.5GHz 的四核 CPU，2GB、4GB 或 8GB 的内存，千兆以太网，Wi-Fi，两个支持 4K 屏幕的 HDMI 端口，一个音频和复合视频输出端口，USB 端口（两个 USB 2.0，两个 USB 3.0），40 个 GPIO 引脚，一个用于树莓派相机模块的摄像头连接器，以及一个 SD 卡插槽。所有这些都集成在一个 88mm×58mm×19.5mm 的板子上，由一个 3A USB-C 电源供电。这些产品起价为 35 美元，比 PC 便宜得多。

🏠 这几款产品的 CPU 都是 ARM 处理器，而不是你在大多数 PC 和 Mac 中常见的英特尔 /AMD x86 或 x64 处理器。这些 CPU 类似于一些微控制器中的 CPU，以及几乎所有的手机、微软 Surface X 和新的基于苹果硅芯片的苹果 Mac 中的 CPU。

Pi Zero

Pi Zero 的体积更小，功率更低。它有一个单核 1GHz CPU，512MB 内存，Wi-Fi（Zero W 型号），一个 HDMI 端口，一个微型 USB 端口，40 个 GPIO 引脚，一个用于树莓派相机模块的相机连接器，以及一个 SD 卡插槽。它的尺寸为 65mm×30mm×5mm，耗电量非常小。Zero 大约是 5 美元，带 Wi-Fi 的 W 版本约 10 美元。

树莓派的所有变种都运行一个名为 Raspberry Pi OS 的 Debian Linux 版本。它有一个没有桌面的精简版，非常适合不需要屏幕展示的“无头”项目，也有一个完整的桌面环境版本，包括网络浏览器、办公应用程序、编码工具和游戏。由于操作系统是 Debian Linux 的一个版本，因此你可以安装任何在 Debian 上运行并为 Pi 内部的 ARM 处理器构建的应用程序或工具。

任务：研究树莓派

如果你在这些课程中使用树莓派，请阅读关于电路板上不同的硬件组件。

- 你可以在树莓派硬件文档 [🔗 \[L2-14\]](#) 页面上找到所使用的处理器的详细信息。了解一下你所使用的树莓派中的处理器。
- 在树莓派硬件文档中找到有关 GPIO 引脚的信息，根据 GPIO 部分的文档 [🔗 \[L2-15\]](#) 来识别你树莓派上的不同引脚。树莓派的 GPIO 引脚定义图如图 2-14 所示。

2.3.2 为单板计算机编程

单板计算机是完整的计算机，运行一个完整的操作系统。这意味着有广泛的编程语言、框架和工具可以用来为它们编码，而不像微控制器那样依赖于 Arduino 等框架对电路板的支持。大多数编程语言都有库，

🔗 做点研究

你熟悉哪些编程语言？它们支持 Linux 吗？

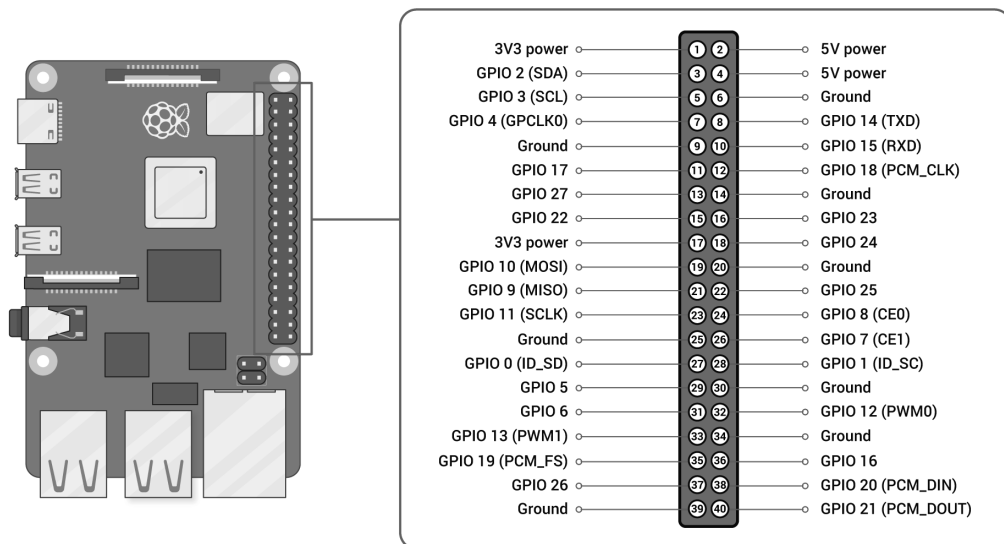


图 2-14 树莓派 GPIO 引脚序号定义图

可以访问 GPIO 引脚，发送和接收来自传感器和执行器的数据。

在树莓派上构建物联网应用最常用的编程语言是 Python。有一个为树莓派设计的巨大的硬件生态系统，而且几乎所有这些硬件都包括使用它们所需的相关代码的 Python 库。其中一些生态系统是基于“hats”（“帽子”）的扩展板，因为它们像一顶帽子一样戴在树莓派的上面，用一个大插座连接到 40 个 GPIO 引脚。这些“帽子”提供了额外的功能，如屏幕、传感器、遥控汽车或适配器，让你用标准化的电缆接入传感器。如图 2-15 所示为树莓派的 Sensor HAT。

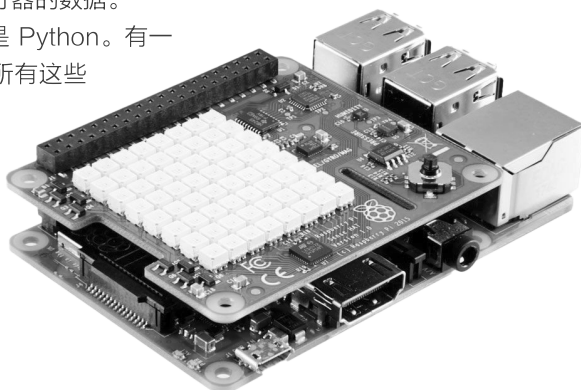


图 2-15 树莓派的 Sensor HAT

2.3.3 在专业物联网部署中使用单板计算机

单板计算机用于专业物联网部署，而不仅仅是作为开发者套件。它们可以提供一种强大的方式来控制硬件和运行复杂的任务，如运行机器学习模型。例如，有一个树莓派 4 计算模块——Raspberry Pi Compute Module 4（也称 CM4），如图 2-16 所示。该模块提供了树莓派 4 的所有功能，但外形紧凑，价格便宜，没有大部分端口，旨在安装到定制硬件中。



图 2-16 树莓派 CM4 计算模块



课后练习



挑战

上一课的挑战是，尽可能多地列出你家里、学校或工作场所的物联网设备。对于清单中的每一个设备，你认为它们是围绕着微控制器还是单板计算机建立的，还是两者的混合？



复习和自学

- 阅读 Arduino 入门指南 [\[L2-16\]](#)，了解更多关于 Arduino 平台的信息。
- 阅读树莓派 4 的介绍 [\[L2-17\]](#)，了解更多关于树莓派的信息。
- 在《电气工程杂志》的“CPU、MPU、MCU 和 GPU 是什么”的文章 [\[L2-18\]](#) 中，了解更多概念和缩略语。

使用这些指南，以及按照硬件指南中的链接所显示的费用，来决定你想使用什么硬件平台，或者你是否愿意使用虚拟设备。

课后测验

(1) 一个 CPU 指令周期的三个步骤包括（ ）。

- A. 译码、执行、提取
- B. 提取、译码、执行
- C. 停止、协作、聆听

(2) 树莓派运行的是什么操作系统？（ ）

- A. 它们不运行操作系统
- B. Windows 95
- C. Raspberry Pi OS

(3) IoT 设备通常比台式计算机和笔记本运行得更快且需要更多的内存。这个说法（ ）。

- A. 正确
- B. 错误



作业

比较和对比微控制器和单板计算机。

说明

本课涉及微控制器和单板计算机。创建一个表格，对它们进行比较，并指出至少两个理由，为什么你会使用微控制器而不是单板计算机；以及至少 2 个理由，为什么你会使用单板计算机而不是微控制器。

评分标准

标准	优秀	合格	需要改进
创建一个比较单片机和单板计算机的表格	创建一个有多个项目的列表，正确地进行比较和对比	创建一个只有几个项目的列表	只能想出一个项目，或没有项目进行比较和对比
使用一个比另一个好的理由	能够为单片机提供两个或更多的理由，为单板计算机提供两个或更多的理由	只能提供 1~2 个关于单片机的理由，和 1~2 个关于单板计算机的理由	不能为单片机或单板计算机提供一个或更多理由

附录 A

本书主页及习题答案

访问本书主页

扫描下方二维码，访问本书的主页，可以获得本书和所需相关硬件的链接，以及书中涉及的链接列表等。

