高等院校计算机应用系列教材

物联网技术基础教程

席 亮 刘 涵 王 珏 主 编

清華大學出版社
北 京

内 容 简 介

本书介绍物联网技术的完整知识体系与工程实践方法。本书共分为15章，深入介绍了物联网的基本概念、发展历程与架构模型，并详细讲解了其核心技术，包括传感器与数据采集、无线通信协议、嵌入式系统开发、边缘计算、云平台集成与数据管理。同时，本书全面探讨了物联网安全与隐私保护、应用系统开发实战、项目管理与部署流程，并拓展介绍了物联网与人工智能的融合(AIoT)、区块链等前沿技术的融合，以及相关的标准化、法规与商业模式。

本书内容丰富、结构清晰、循序渐进，注重理论联系实际，各章均配有实例与思考练习题，最后一章还专门提供了综合项目设计指导与职业发展建议。本书主要面向物联网工程、计算机科学与技术等相关专业的本科生，适合作为高等院校相关课程的教材，也可供物联网领域初学者和工程技术人员参考。

本书配套的电子课件、教案、习题答案和实例源文件，扫描封底或前言中的二维码即可获取。

图书在版编目(CIP)数据

物联网技术基础教程 / 席亮, 刘涵, 王珏主编.
北京 : 清华大学出版社, 2026. 7. -- (高等院校计算机应用系列教材). -- ISBN 978-7-302-72362-2

Ⅰ. TP393.4；TP18

中国国家版本馆CIP数据核字第2026C60L84号

责任编辑：胡辰浩
封面设计：高娟妮
版式设计：妙思品位
责任校对：马遥遥
责任印制：刘　菲

出版发行：清华大学出版社
网　　址：https://www.tup.com.cn，https://www.wqxuetang.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：大厂回族自治县彩虹印刷有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张：**25.25　　**字　　数：**646千字
版　　次：2026年8月第1版　　**印　　次：**2026年8月第1次印刷
定　　价：79.80元

产品编号：113232-01

前　言

我们正处在一个由深度数字化和万物互联驱动的智能时代。信息技术的每一次飞跃都在重塑社会的运行模式，深刻改变着人类的生活、工作与协作方式。在构成现代信息技术大厦的诸多支柱中，物联网技术与人工智能、大数据、云计算等共同扮演着核心引擎的角色。自概念诞生以来，物联网技术不断突破，将物理世界的“物”与数字世界的“网”无缝融合，使得感知、连接和智能控制无处不在。在当今热门的智慧城市建设、工业互联网升级、智能家居普及、精准农业实施及数字医疗发展等领域，物联网都已成为不可或缺的底层支撑。

作为一门综合性极强的交叉学科，物联网起步于感知与连接，但其内涵与边界正在快速扩展。它不仅是“联网的设备”，更是一个集成了传感、通信、计算、平台与智能应用的庞大生态系统。由于物联网技术持续融合前沿成果，以解决各行各业日益增长和变化的数字化、智能化需求，其技术栈日益丰富，应用部署越来越便捷，系统的可靠性与安全性也越来越受到重视，从而使该技术的应用呈现爆发式增长。在我国，物联网的应用已深入智能制造、智慧能源、智慧交通、环境监测、公共安全、健康养老等众多行业和领域，成为推动产业转型升级和数字经济高质量发展的重要力量。物联网为千行百业提供了端到端的智能化解决方案，助力企业构建灵活、高效、可感知的运营体系，增强其对市场变化的敏捷响应能力，从而提升核心竞争力。

本书从物联网的基本概念与体系架构出发，由浅入深地详细讲述了其核心技术组成部分，包括传感器与数据采集、无线通信协议、嵌入式系统开发、边缘计算、物联网云平台与数据管理，并设专章深入探讨了物联网安全与隐私保护这一生命线。本书不仅系统阐述物联网从感知层到应用层的完整技术链条，还通过智能家居、智慧农业等综合实战章节，培养读者解决实际工程问题的能力。更进一步，本书前瞻性地探讨了人工智能与物联网的融合、区块链应用及数字孪生等前沿趋势，并对项目管理、标准化、商业模式与职业发展进行了梳理，旨在帮助读者构建系统性的知识框架并把握未来方向。

本书内容丰富、结构合理、思路清晰、语言简练流畅、示例翔实。每一章均以学习目标为引导，在正文中结合关键技术与难点，穿插了大量贴近实际应用的示例。每一章末尾都安排了有针对性的思考与练习题，有助于巩固基本概念并培养动手实践能力。本书最后特别设置了“课程总结与项目实践”章节，引导读者完成从选题、设计到实现的全过程，将分散的知识点融会贯通。

本书主要面向物联网工程、计算机、电子信息等相关专业的初学者，适合作为高等院校相关课程的教材及物联网技术培训班的培训资料，也可供广大物联网应用开发工程师和爱好者参考。

本书由席亮(哈尔滨理工大学)、刘涵(哈尔滨理工大学)和王珏(齐齐哈尔大学)合作编写，其中席亮编写第2、3、5、11、12、15章，刘涵编写第1、4、6、7、14章，王珏编写第8、9、10、13章。

由于作者水平有限，书中难免存在不足之处，恳请专家和广大读者批评指正。编写本书的过程中参考了相关文献，在此向这些文献的作者深表感谢。联系方式：电话010-62796045，电子邮箱992116@qq.com。

本书配套的电子课件、教案、习题答案和实例源文件扫描下方的二维码即可获取。

作　者

2026年3月

目　录

第 1 章

物联网概述

本章导读

物联网(Internet of Things，IoT)是近年来信息技术领域的重要分支之一，其理念是通过将物理世界中的各种设备、物体和系统连接到互联网，实现智能化的感知、通信、分析和控制。随着传感器技术、通信技术、人工智能和大数据处理能力的不断提升，物联网已渗透到各行各业，从智能家居到智慧城市，从工业自动化到医疗健康，表现出巨大的应用潜力和市场前景。

本章首先介绍物联网的定义、与互联网的区别、发展历程、典型的三层架构模型；随后讲解物联网的关键技术，包括传感器、网络通信、数据处理与存储，以及安全与隐私保护；接着介绍物联网的行业应用，包括智能家居、工业物联网、智慧城市和医疗健康物联网；最后介绍国内外物联网产业现状，探讨物联网与5G、AI、大数据等技术的关系，并展望未来趋势。

通过本章的学习，读者将对物联网有一个全面的认识，理解其技术基础、应用场景以及未来发展方向，为后续深入学习物联网相关技术与应用打下基础。

本章学习目标

- 理解物联网的定义、特点、与互联网的区别，了解物联网的发展历程与发展现状。
- 能准确区分物联网三层架构的功能边界，熟悉传感器、网络通信、数据存储、数据处理、安全与隐私保护等技术的基本原理和应用。
- 了解物联网在智能家居、工业物联网、智慧城市和医疗健康等领域的应用。
- 了解国内外物联网发展现状，理解物联网与5G、人工智能、大数据等技术的融合关系，并展望未来的发展趋势。

1.1 物联网的基本概念

1.1.1 什么是物联网 (IoT)

物联网是指通过互联网将物理世界中的各种物体、设备、系统相互连接，使其具备感知、识别、通信、计算和控制的能力，从而实现智能化管理和服务的一种新型网络体系。物联网的理念在于“物物相连”，即通过传感器、通信模块、数据处理单元等技术方式，

使现实世界中的物体能够被远程感知、监控和管理。

物联网的基本特征如下。

- 感知化：通过传感器等设备对环境、物体状态等进行实时感知。
- 互联化：通过无线或有线通信技术把物体接入互联网，实现信息的交互与共享。
- 智能化：利用数据分析、人工智能等技术对获取的信息进行处理，实现自主决策和智能控制。

物联网应用广泛，主要与各行业相结合，包括智能家居、智慧城市、工业自动化、医疗健康、农业监测、交通管理等多个领域。例如，在智能家居中，用户可以通过手机远程控制家中的灯光、空调、安防系统等设备；在智慧交通中，车辆可以通过物联网技术实现自动驾驶、交通流量优化等功能。

1.1.2 物联网与互联网的区别

物联网与互联网都涉及网络连接和信息传输，但是它们在概念、技术架构和应用方面有着明显的区别，如表1-1所示。

表 1-1 物联网与互联网的区别

比较项	互联网	物联网
连接对象	人与人之间的信息交互为主	物与物、物与人之间的信息交互为主
主要功能	信息采集、传输与共享	物体的感知、控制与智能决策
通信方式	主要是有线或无线网络，如Wi-Fi、以太网	包括短距离通信(如蓝牙、ZigBee)和广域网通信(如NB-IoT、LoRa)
数据类型	文字、图像、视频等多媒体信息	传感器采集的实时数据，如温度、湿度及地理位置等
智能程度	依赖人工操作与判断	具备一定的自主感知、分析和决策能力
应用场景	网络社交、电子商务、在线办公等	智能家居、智慧交通、工业监控、医疗健康

简单来说，互联网是以人为中心的信息网络，而物联网是以物为中心的感知网络；互联网主要解决信息的获取与传递问题，而物联网更强调对物理世界的实时感知与智能控制。

1.1.3 物联网的发展历程

物联网的发展可以追溯到20世纪末。物联网经历了从概念提出到逐步落地的过程，主要分为以下几个关键阶段。

1. 概念提出阶段(20世纪80年代—1999年)

这是“物物互联”思想的提出和相应技术的早期探索阶段。1983年，美国卡内基梅隆大学将一台可乐贩卖机接入网络，以实现通过网络查询机器内可乐库存与温度的功能，被视为物联网的雏形。1999年，美国麻省理工学院(MIT)自动识别中心首次提出“物联网”的概念，即“通过RFID技术实现物品自动识别与信息共享的网络”，明确了RFID是物联网的主要技术方向之一。

2. 技术探索阶段(2000—2008年)

RFID技术及传感器技术的进步，以及互联网的普及，推动物联网进入技术积累期。

2003 年，国际电信联盟(ITU)发布《互联网报告 2005：物联网》，正式将物联网纳入全球信息通信技术发展框架。Wi-Fi、蓝牙等无线通信技术逐步商业化，嵌入式系统的性能得到大幅提升，为物联网的规模化应用奠定了技术基础，但本阶段尚未形成成熟的产业生态。

3. 标准制定与产业推广阶段(2009—2019 年)

本阶段物联网进入标准化发展时期。国际标准化组织(ISO)、国际电信联盟(ITU)等机构陆续推出物联网标准。2009 年，物联网被正式列为中国战略性新兴产业。各国纷纷出台政策推动物联网发展，物联网进入快速落地期。该阶段，LoRa、NB-IoT等低功耗广域网技术逐步商用，解决了当时物联网通信设备的痛点(如远距离、低功耗通信等)；云计算的兴起为物联网产生的海量数据提供了存储与处理能力；智能家居、工业监控等场景开始规模化应用，物联网产业生态初步形成。2015 年，阿里云、华为云等企业推出物联网平台，进一步降低了设备接入与应用开发的门槛。

4. 广泛应用与融合发展阶段(2020 年至今)

随着 5G、人工智能及大数据技术的成熟，物联网进入“AIoT(人工智能+物联网)”融合发展阶段。5G技术的高带宽、低延迟、广连接特性，满足了高清视频监控、远程控制等复杂物联网场景的需求；人工智能技术与物联网融合，实现了从“数据采集”到“智能决策”的升级，例如，基于AI技术的设备故障预测、智能交通调度；数字孪生、区块链等新技术与物联网融合，拓展了应用边界。上述技术的应用，形成了覆盖全行业的智能化解决方案，物联网正式进入“万物智联”时代。

1.1.4　物联网的三层架构模型

为了更好地理解和设计物联网系统，业界普遍采用三层架构模型，即感知层、网络层和应用层，如图 1-1 所示。每一层都有其特定的功能和关键技术，共同构成完整的物联网体系。

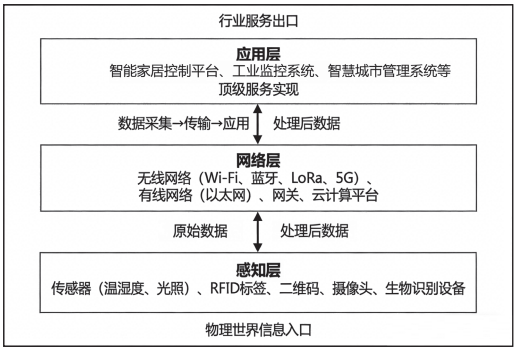

图 1-1　物联网三层架构模型

1. 感知层

感知层是物联网的最底层，负责对物理世界的信息进行采集和识别。例如，采集物理世界的各类信息，如环境参数、设备状态及位置信息等，并将其转化为可传输的数字信号。该层主要由各种传感器(如温湿度传感器、光照传感器等)、RFID标签、摄像头、GPS定位设备等组成。其核心功能如下。

- 数据采集：通过传感器实时获取温度、湿度、光照、位置等环境数据。
- 身份识别：利用RFID、二维码、NFC等技术对物体进行唯一标识。
- 数据预处理：对采集的数据进行初步处理，如滤波、压缩、格式转换等。

感知层的关键技术如下。

- 传感器技术：用于检测物理量并将其转换为电信号。
- RFID技术：通过无线信号识别物体的身份信息。

- 二维码/NFC技术：实现快速识别和数据交换。

例如，在智慧农业场景中，感知层通过土壤湿度传感器采集土壤数据，通过RFID标签标识农作物信息，为后续处理提供基础数据。

2. 网络层

网络层是物联网的传输层，负责将感知层采集的数据传输到上层应用系统。该层包含各种通信技术和网络协议，确保数据的高效、可靠传输。网络层的主要功能如下。

- 数据传输：通过无线或有线网络将数据从感知层传送到应用层。
- 路由选择：根据网络拓扑结构和通信需求选择最优路径。
- 网络管理：实现设备的接入控制、流量管理、安全防护等。

网络层的关键技术如下。

- 无线通信技术：如Wi-Fi、蓝牙、ZigBee、NB-IoT、LoRa、5G等。
- 有线通信技术：如以太网、光纤通信等。
- 网络协议：如TCP/IP、MQTT、CoAP、HTTP等。

例如，边缘网关可将感知层设备采集的多源数据进行汇聚、预处理之后，通过5G网络传输至云端平台，提升数据传输效率与可靠性。

3. 应用层

应用层是物联网的最上层，负责对网络层传输的数据进行存储、分析及处理，并将其转化为针对具体场景的智能决策与应用服务。该层直接面向用户，提供各种智能化应用，如智能家居控制、工业监测、城市管理、医疗健康等。应用层的主要功能如下。

- 数据分析与处理：利用大数据、人工智能等技术对海量数据进行挖掘和分析。
- 智能决策：基于数据分析结果，实现自动化控制和优化决策。
- 用户交互：提供可视化界面，方便用户进行远程监控和操作。

应用层的关键技术如下。

- 云计算与边缘计算：提供强大的数据处理和存储能力。
- 人工智能：实现智能识别、预测和决策。
- 用户接口技术：如移动应用、Web平台、语音交互等。

1.2 物联网的关键技术

物联网是一门交叉学科，融合了传感技术、通信技术、计算机技术、人工智能技术等多领域成果。其中，感知层、网络层、应用层各自对应的核心技术，共同构成了物联网的技术体系。以下重点介绍四大关键技术：传感器技术、网络通信技术、数据处理与存储技术、安全与隐私保护技术。

1.2.1 传感器技术

传感器是物联网感知层的核心设备，用于将物理世界中的各种信息(如温度、湿度、压力、光照、位置等)转换为电信号，供后续处理和分析。传感器技术的发展直接影响物联网的感知能力和应用范围。

1. 传感器的分类

根据测量对象的不同，传感器可分为以下几类。

- 温度传感器：用于测量环境温度，广泛应用于智能家居、工业监测等领域。
- 湿度传感器：用于检测空气中的湿度，常见于农业、气象监测等场景。
- 压力传感器：用于测量气体或液体的压力，常见于汽车、航空航天等领域。
- 光学传感器：用于检测光照强度，应用于智能照明、安防监控等。
- 加速度传感器：用于测量物体的加速度，常见于智能穿戴设备、无人机等。
- GPS传感器：用于获取物体的地理位置信息，广泛应用于物流、车联网等领域。

2. 传感器的发展趋势

随着物联网技术的发展，传感器技术正朝着微型化、低功耗、高精度、多参数融合的方向演进。例如，微型传感器可嵌入微小设备，如智能手表、医疗植入设备；低功耗传感器能满足电池供电设备的长期运行需求，如户外环境监测设备；多参数传感器可同时采集多种物理量，减少设备部署成本，提升数据关联性。

1.2.2　网络通信技术

网络通信技术是物联网数据传输的核心，决定了物联网系统的覆盖范围、传输速率、能耗和安全性。其技术选型需结合物联网场景的特点，如传输距离、功耗、带宽需求。

根据通信距离的不同，物联网通信技术可以分为短距离通信和广域网通信两类。

1. 短距离通信技术

短距离通信适用于局域网内的设备互联，常见技术如下。

- Wi-Fi：高速无线通信技术，适用于智能家居、办公网络等场景。
- 蓝牙(Bluetooth)：低功耗短距离通信技术，广泛应用于可穿戴设备、无线耳机等。
- ZigBee：低功耗、低成本的无线通信协议，适用于工业自动化、智能家居等。
- NFC(近场通信)：用于短距离非接触式数据交换，常见于移动支付、门禁系统等场景。

2. 广域网通信技术

广域网通信的传输距离可达几公里至几十公里，功耗极低，适用于户外、长待机的物联网场景。常见技术如下。

- NB-IoT(窄带物联网)：基于蜂窝网络的低功耗广域网技术，适用于智能电表、环境监测等。
- LoRa：远距离、低功耗通信技术，适用于智慧农业、城市照明等。
- LTE-M (LTE-Machine)：支持移动性和低功耗，适用于物流追踪、智能穿戴等。

3. 蜂窝技术

依托运营商的移动网络(4G、5G)，具有传输距离远、带宽高、移动性强的特点，适用于需要高速传输数据或移动场景的物联网应用。常见技术如下。

- 4G：用于车载物联网、远程视频监控。
- 5G：凭借低延迟(毫秒级)、广连接(每平方公里百万级连接)的优势，支撑工业远程控制、智能交通、车联网等复杂场景。

4. 网络协议

物联网通信还需要一系列网络协议来保证数据的可靠传输，主要涉及以下内容。

- MQTT (消息队列遥测传输)：轻量级发布/订阅协议，适用于低带宽及不稳定的网络环境。
- CoAP (受限应用协议)：专为受限设备设计的协议，适用于资源有限的物联网设备。
- HTTP/HTTPS：传统Web协议，适用于需要与Web服务交互的物联网应用。

1.2.3 数据处理与存储技术

物联网系统运行过程中会产生海量、多源、异构的数据，如传感器实时数据、设备日志数据、用户交互数据。这类数据具有“实时性、碎片化、高并发”的特点，传统数据处理与存储技术难以满足其需求。因此，如何高效地处理和存储这些数据是物联网应用的关键挑战之一，并催生了以下针对性的技术方案。

1. 数据存储技术

针对物联网海量时序数据(即传感器按时间序列采集的数据)，可采用时间序列数据库(如InfluxDB、TDengine)。这类数据库能高效存储和查询时序数据，支持数据按时间范围检索及压缩存储，降低存储成本。同时，云计算技术提供了弹性存储服务，物联网平台可根据数据量动态调整存储资源，满足不同场景的存储需求。

2. 数据处理技术

数据处理可分为边缘计算与云计算两种模式。边缘计算是在靠近数据采集端(边缘节点)进行数据预处理，过滤冗余数据并提取关键信息，再将数据传输至云端。该模式能降低网络带宽压力并减少数据传输延迟，适用于实时控制场景(如工业设备远程控制)。云计算则通过云端服务器对海量数据进行集中分析、挖掘，利用大数据技术(如Hadoop、Spark)处理大规模数据，并结合人工智能技术实现智能决策(如用户行为分析、设备故障预测)。

数据处理与存储技术的核心目标是“从海量数据中提取价值”，为物联网应用层的智能服务提供支撑，是物联网实现“数据驱动”的关键。

1.2.4 安全与隐私保护技术

物联网连接海量物理设备，涉及个人隐私、工业机密、公共安全等敏感信息，其安全与隐私保护问题尤为突出。设备被入侵、数据被篡改、隐私信息泄露等风险可能引发严重后果。因此，安全与隐私保护技术是物联网不可或缺的关键技术，主要涵盖以下4个方面。

(1) 设备安全：物联网设备多为嵌入式设备，算力较弱且防护能力较差，易被攻击。设备安全技术通过固件加密、安全启动、身份认证等方式，防止设备被非法入侵与控制。例如，对设备固件进行数字签名，确保固件不被篡改；采用设备唯一标识(Device ID) +密钥的方式，验证设备接入的合法性。

(2) 通信安全：保障数据传输过程中的机密性与完整性，防止数据被窃听、篡改。核心技术包括加密技术(对称加密AES、非对称加密RSA)及安全通信协议(TLS/SSL、MQTT-SN)等。例如，通过TLS/SSL协议对物联网设备与云端之间的传输数据进行加密，确保数据传输安全。

(3) 数据安全：数据安全旨在保护存储与处理过程中的数据安全，防止数据泄露、篡改与滥用。核心技术包括数据加密存储、访问控制、数据脱敏等。例如，对个人位置信息、医疗数据等敏感数据进行加密存储，设置严格的访问权限，仅授权用户可访问；对公开数据进行脱敏处理，删除隐私字段。

(4) 管理安全：通过建立完善的安全管理体系，实现对设备、数据、用户的全生命周期安全管控。包括密钥管理、日志审计、入侵检测与响应等。例如，建立密钥生命周期管理系统，定期更新密钥；通过日志审计追溯数据访问与操作记录，及时发现异常行为。

随着物联网应用的普及，安全与隐私保护已成为影响行业发展的关键因素，相关技术也在不断迭代，以应对日益复杂的安全威胁。

1.3 物联网的应用领域

物联网技术凭借“万物互联、智能赋能”的核心优势，已渗透到社会生产生活的方方面面，颠覆了传统行业的运行模式，并催生了新的产业生态。以下重点介绍四大主流应用领域：智能家居、工业物联网(IIoT)、智慧城市和医疗健康物联网。

1.3.1　智能家居

智能家居是物联网最早和最广泛的应用之一。通过将家庭中的各种设备(如灯光、空调、电视、安防系统、智能门锁等)连接到互联网，用户可以通过手机、语音助手或智能中控系统实现远程控制和自动化管理，提升家庭生活的舒适性、便捷性与安全性。

典型应用场景包括：智能照明系统根据室内光照强度与人体感应自动开关、调节亮度；智能安防系统通过摄像头、门磁传感器、烟雾报警器，实现远程监控、异常报警(如有人闯入、燃气泄漏)，用户可通过手机App实时查看家庭状态并远程处理；家电联动系统实现“场景化控制”，如“起床模式”自动开启窗帘、打开灯光、煮好咖啡，“睡眠模式”自动关闭灯光、关闭家电电源并锁好门窗。

智能家居的核心技术支撑包括短距离通信技术(Wi-Fi、蓝牙)、传感器技术、智能控制算法与用户交互技术。目前已形成成熟的产品生态，从单一智能设备(如智能音箱、智能灯泡)，向全屋智能解决方案演进。

1.3.2　工业物联网 (IIoT)

工业物联网(Industrial Internet of Things，IIoT)是物联网技术与工业生产深度融合的产物，其核心是通过连接工业设备、生产线、供应链等要素，实现生产过程的智能化监测、控制、优化与管理，推动传统制造业向“智能制造”转型。

典型应用场景包括：设备状态监测与预测性维护——通过在工业设备上安装振动传感器、温度传感器，实时采集设备运行数据，结合AI算法分析设备状态，预测故障风险并提前预警，减少非计划停机时间；智能生产线——通过物联网技术实现设备、物料、人员的协同联动，自动调度生产流程，提升生产效率与产品质量；供应链智能化管理——通过RFID技术标识物料与产品，实时追踪物料运输、仓储、加工全流程，实现供应链可视化与高效管控。

工业物联网对技术的稳定性、实时性、安全性要求较高，核心依赖5G、边缘计算、工业以太网等技术，是物联网产业中产值规模最大、技术门槛最高的领域之一。

1.3.3 智慧城市

智慧城市是物联网技术在城市治理与公共服务领域的规模化应用，主要通过连接城市中的交通、能源、环保、安防、医疗等公共设施，实现城市运行状态的实时感知、智能调度与高效管理，提升城市治理水平与居民生活质量。

典型应用场景包括：智能交通——通过摄像头、地磁传感器、GPS等设备采集交通流量、车辆位置数据，动态调节交通信号灯，优化行车路线，缓解交通拥堵；智能能源——通过物联网技术监测电网、供水、供气系统的运行状态，实现能源精准调度、泄漏检测与节能优化；智能环保——通过部署空气质量传感器、水质传感器、噪声传感器，实时监测城市环境质量，及时预警污染事件，辅助环保决策；智能安防——构建城市级视频监控网络与应急响应系统，实现对违法犯罪、突发事件的快速识别与处置。

智慧城市是一个复杂的系统工程，需整合物联网、5G、AI、大数据等多种技术，涉及政府、企业、居民等多方主体，是当前各国数字城市建设的核心方向。

1.3.4 医疗健康物联网

医疗健康物联网(Internet of Medical Things, IoMT)是物联网技术与医疗健康领域的融合应用，主要通过智能医疗设备、可穿戴设备、远程监测系统，实现患者健康数据的实时采集、远程诊断、精准治疗与健康管理，推动医疗服务从“院内诊疗”向“院外健康管理”延伸。

典型应用场景包括：远程患者监测——通过可穿戴设备(如智能手环、血糖监测仪)实时采集患者心率、血压、血糖等健康数据，并传输至医院云端平台，医生可远程监控患者状态，及时调整治疗方案，尤其适用于慢性病患者、老年患者；智能医疗设备——如物联网血糖仪、心电图机，可自动记录检测数据并同步至医疗系统，减少人工记录误差，提升诊疗效率；医院智能化管理——通过物联网技术实现医疗设备(如呼吸机、输液泵)的定位与状态监测，优化设备调度；药品追溯——通过RFID技术实现药品从生产、仓储、运输到使用的全流程追溯，保障用药安全。

医疗健康物联网直接关系到人体健康与生命安全，对数据准确性、安全性、隐私保护的要求极高，同时依赖AI辅助诊断、5G远程医疗等技术的支撑，是物联网应用中极具社会价值的领域。

1.4 物联网发展现状与趋势

1.4.1 全球物联网市场规模

近年来，全球物联网产业保持高速增长态势，得益于5G、AI、边缘计算等技术的成熟，以及各行业数字化转型需求的释放，物联网市场规模持续扩大。根据市场研究机构IDC的数据，2025年全球物联网市场规模达到1.1万亿美元，较2020年增长近一倍；全球物联网连接设备数量预计突破750亿台，涵盖消费电子、工业、交通、医疗等多个领域。

从市场结构来看，工业物联网、智慧城市、智能家居是全球物联网市场的核心增长点。其中，工业物联网凭借制造业数字化转型的刚需，占比最高，预计2025年其全球市场规模占比超过30%；智慧城市领域受各国政策推动，增长速度最快，尤其在亚洲、欧洲等地

区，智慧城市项目密集落地，带动物联网设备与解决方案需求增长；智能家居市场则依托消费升级需求，保持稳定增长，智能音箱、全屋智能解决方案等产品渗透率持续提升。

从区域分布来看，北美、欧洲、亚太地区是全球物联网市场的主要阵地。北美地区凭借技术研发优势与成熟的产业生态，在物联网芯片、云平台、AI算法等领域占据领先地位；欧洲地区聚焦智慧城市与工业物联网，出台了多项政策推动物联网标准化与规模化应用；亚太地区凭借庞大的人口基数、制造业基础与政策支持，成为全球物联网市场增长最快的区域，中国、日本、韩国是核心驱动力。

1.4.2　中国物联网产业发展现状

中国物联网产业在政策支持、技术积累、市场需求的多重驱动下，实现了快速发展，已形成从芯片、设备、通信到平台、应用的完整产业链，产业规模连续多年保持两位数增长。根据中国信通院的数据，2024年中国物联网产业规模突破3.5万亿元，物联网连接设备数量超过300亿台，均位居全球前列。

从产业链布局来看，中国物联网产业在中上游设备制造、下游应用场景领域具有明显优势。中上游方面，中国是全球最大的物联网终端设备生产国，在传感器、通信模组、嵌入式终端等领域形成了规模化产能，华为、移远通信等企业在通信模组领域全球市场份额领先；下游应用方面，智能家居、工业物联网、智能抄表等场景的应用渗透率位居全球前列，阿里云、华为云、腾讯云等企业的物联网平台，已成为全球主流的物联网设备接入与管理平台，支撑海量设备与场景的智能化需求。

政策支持是中国物联网产业发展的重要保障。国家先后出台《物联网发展规划(2021—2025年)》《“十四五”数字经济发展规划》等政策，将物联网作为战略性新兴产业予以重点扶持，明确了技术研发、场景应用、标准化、安全保障等发展方向。同时，地方政府也纷纷布局物联网产业园区，推动产业链上下游协同发展，形成了长三角、珠三角、京津冀等物联网产业集聚区域。

目前，中国物联网产业仍面临部分核心技术“卡脖子”问题，如高端传感器、物联网芯片、核心操作系统等领域与国际领先水平存在差距，产业链自主可控能力有待提升。但随着国产替代进程加快与技术研发投入增加，这些问题正逐步得到解决。

1.4.3　物联网与 5G、AI、大数据的关系

物联网并非孤立存在，其发展与5G、AI、大数据等新兴技术深度融合、协同赋能，形成了“1+N”的技术生态。其中，物联网是主要载体，5G、AI、大数据是关键支撑，共同推动“万物智联”的实现。

1. 物联网与5G

物联网与5G相互支撑，拓展应用边界。5G技术为物联网提供了高性能的通信支撑，解决了传统物联网技术在带宽、延迟、连接数方面的瓶颈。5G的高带宽特性可满足高清视频监控、VR/AR等物联网场景的需求；低延迟特性可支撑工业远程控制、车联网等实时性要求高的场景；广连接特性可实现海量物联网设备的同时接入，为智慧城市、工业物联网等大规模场景提供可能。同时，物联网也为5G提供了丰富的应用场景，带动5G网络建设与商业化落地，二者形成“技术支撑—场景落地”的良性循环。

2. 物联网与AI

物联网与AI结合，通过数据驱动，实现智能升级。物联网的核心价值在于采集海量数据，而AI的核心能力在于从数据中提取价值、实现智能决策，二者的融合形成了AIoT(人工智能+物联网)，这是物联网发展的主要方向。物联网为AI提供了丰富的真实场景数据，作为AI模型训练与优化的基础；AI技术则对物联网采集的海量数据进行分析、挖掘，将数据转化为智能决策指令，实现从“感知”到“认知”再到“决策”的升级。例如，物联网采集的工业设备振动数据，通过AI算法分析可预测设备故障；物联网采集的交通数据，通过AI调度算法可优化交通流量。

3. 物联网与大数据

物联网与大数据结合，可实现协同处理，挖掘数据价值。物联网是大数据的重要数据来源，其设备采集的多源、异构、时序数据构成了大数据的核心数据源之一；大数据技术则为物联网数据提供了高效的处理与分析方案，解决了物联网“数据海量、价值稀疏”的问题。大数据技术通过数据清洗、汇聚、挖掘等方式，从海量物联网数据中提取有价值的信息，为物联网应用层的智能服务提供支撑。例如，通过大数据分析用户的智能家居使用习惯，可个性化推荐智能场景模式；通过大数据分析农业物联网采集的环境数据，可优化种植方案，提升农作物产量。

综上，5G、AI、大数据与物联网的融合并非简单的技术叠加，而是形成了“感知→传输→处理→决策”的完整闭环，推动物联网从“万物互联”向“万物智联”跨越。

1.4.4 物联网未来发展趋势

随着技术进步与场景渗透的不断深入，物联网未来将呈现出“智能化、融合化、标准化、安全化”的发展趋势，同时拓展出更多新兴应用场景，重构产业生态。

(1) AIoT深度融合，智能化水平持续提升。AI与物联网的融合将成为主流趋势，AI技术将全面渗透到物联网的感知层、网络层及应用层。感知层将出现具备AI推理能力的智能传感器，可实现数据的本地智能分析与筛选；网络层将通过AI优化通信资源调度，提升数据传输效率与可靠性；应用层将通过AI实现更精准的智能决策与个性化服务。未来，物联网设备将从“被动响应”转变为“主动感知、智能决策”，实现真正的“万物智联”。

(2) 多技术交叉融合，应用边界不断拓宽。除了与5G、AI、大数据的融合，物联网还将与数字孪生、区块链、量子通信等新技术深度融合，拓宽应用边界。例如，物联网与数字孪生结合，可构建物理设备的虚拟映射模型，实现设备全生命周期的可视化管理与仿真优化，广泛应用于工业、智慧城市领域；物联网与区块链结合，可实现设备身份认证、数据不可篡改，提升物联网系统的安全性与信任度，适用于供应链、医疗健康等场景；物联网与量子通信结合，将从根本上解决通信安全问题，支撑高安全需求的物联网场景(如金融、国防)。

(3) 标准化体系逐步完善，产业协同加速。目前，物联网行业存在多协议并存、接口不统一等问题，制约了产业协同发展。未来，国内外将加快物联网标准化建设，推动通信协议、数据格式、接口规范、安全标准的统一，实现不同企业、不同设备、不同场景之间的互联互通。标准化将降低设备接入与应用开发成本，促进产业链上下游协同创新，推动物联网产业规模化、规范化发展。

(4) 安全与隐私保护常态化，技术持续升级。随着物联网应用的普及，安全与隐私保护将成为行业发展的底线，相关技术将持续迭代升级。一方面，安全技术将从“被动防护”

向“主动防御”转变，通过AI入侵检测、区块链数据存证等技术，实现安全风险的实时预警与处置；另一方面，隐私保护将更加注重合规性，结合数据安全法规，建立完善的数据收集、存储、使用、销毁全流程隐私保护机制，平衡技术创新与隐私保护。

(5) 新兴场景加速落地，产业生态持续丰富。物联网将从当前的主流场景(如智能家居、工业物联网、智慧城市)向更细分、更前沿的场景延伸，如元宇宙物联网(虚拟世界与物理世界的联动)、低空经济物联网(无人机、低空飞行器的智能管控)、海洋物联网(海洋环境监测、远洋航运管理)等。同时，物联网将催生新的商业模式，如设备即服务(DaaS)、数据即服务(Data as a Service)，推动产业生态从“硬件销售”向“服务赋能”转型。

1.5 本章小结

本章围绕物联网的核心知识展开。首先明确了物联网的定义、核心特征与发展历程，阐述了物联网与互联网的本质区别，构建了物联网的基础认知；随后介绍了物联网三层架构模型(感知层、网络层、应用层)，剖析了各层的功能定位与核心技术；接着重点讲解了传感器技术、网络通信技术、数据处理与存储技术、安全与隐私保护技术四大关键技术，以及智能家居、工业物联网、智慧城市、医疗健康物联网四大主流应用领域；最后分析了全球与中国物联网产业发展现状，探讨了物联网与5G、AI、大数据的融合关系及未来发展趋势。

通过本章学习，读者应建立对物联网的整体认知框架，理解物联网“万物互联、数据驱动、智能赋能”的核心逻辑，掌握三层架构的核心内涵与关键技术的应用场景，同时认清物联网产业的发展现状与未来方向，为后续章节中核心技术、实战应用的深入学习奠定基础。

1.6 思考与练习

一、选择题

1. 物联网三层架构中，负责数据采集与信号转换的是(　　)。

A. 感知层　　B. 网络层　　C. 应用层　　D. 平台层

2. 下列哪项技术不属于物联网短距离无线通信技术？(　　)

A. Wi-Fi　　B. NB-IoT　　C. 蓝牙　　D. ZigBee

二、简答题

1. 简述物联网与互联网的核心区别，并结合具体场景说明二者的协同关系。

2. 物联网三层架构各自的核心功能是什么？请举例说明各层技术在智能家居场景中的应用。

3. 结合本章内容，分析AI、5G技术对物联网发展的赋能作用，并举例说明AIoT在工业或医疗领域的应用前景。

三、实践题

观察身边的物联网应用场景(如智能快递柜、智能电表、智能手环等)，分析其背后涉及的物联网关键技术、架构组成，撰写一份不少于500字的分析报告。

第 2 章

物联网体系架构

本章导读

上一章讲解了物联网的整体框架，包括感知层、网络层、应用层。本章将在此基础上，进一步把应用层拆分为“平台与云服务层、应用与用户交互层”，同时将原来的“网络层”明确定位为“网络传输层”，由此形成物联网的四层架构：“感知层、网络传输层、平台与云服务层、应用与用户交互层”。随后，本章聚焦各层级的核心设备、关键技术与实操实例，进一步拆解物联网系统的组成：从感知层的传感器与嵌入式设备采集数据，到网络传输层的多协议通信链路搭建，再到云平台的设备管控与数据存储，最终通过应用层实现用户交互与智能服务，层层递进地讲解各环节的协同逻辑。通过理论讲解与示例实践，帮助读者掌握物联网体系架构、各层的硬件选型、网络搭建、平台接入与界面开发等基础知识，为后续核心技术学习与项目实战夯实工程能力。

本章学习目标

- 掌握物联网四层架构各层级的核心设备、技术原理与功能定位，熟记主流传感器、通信协议、云平台的特性，理解设备管理、消息队列、用户交互及数据可视化的核心机制。
- 能根据场景需求选择传感器与通信技术，理解基于Arduino、ESP32的基础开发，了解设备接入云平台及简易物联网监控Web界面的开发方法，理解各层级数据流转逻辑与用户行为分析方法。
- 建立物联网系统的整体工程思维，培养硬件与软件协同设计的意识，形成“需求→选型→开发→落地→优化”的实操闭环思维。

2.1 物联网体系架构概述

物联网是一个层次化的网络。物联网大致分为3层，分别为感知层、网络层和应用层。本书从技术角度将应用层进一步拆分为平台与云服务层及应用与用户交互层，从而形成四层物联网体系架构：感知层、网络层、平台与云服务层、应用与用户交互层。各层次涉及的关键技术较多，是典型的跨学科技术。物联网通过对现有技术的综合运用来实现全新的通信模式。在对现有技术的融合中，物联网提出了对现有技术的改进和提升要求，并催生出新的技术体系。

物联网的体系结构如图2-1所示，从下到上依次划分为感知层、网络层、平台与云服务层、应用与用户交互层。各层之间，信息不是单向传递的，也存在交互或控制关系。所传递的信息中，主要是物的信息，包括物的识别码、物的静态信息及动态信息等。

图 2-1　物联网四层体系架构

物联网(IoT)四层体系架构通过新增平台与云服务层，实现了设备管理、数据处理与应用开发的专业化分工，使整个物联网系统更具扩展性、可维护性和智能化水平。各层相互支撑、协同工作，构成完整的物联网运行体系。

1. 感知层

感知层是物联网的“感知器官”，是实现“万物互联”的基础。感知层是物联网体系的最底层，也是整个系统的数据采集源头，核心作用是“感知万物、采集数据”，相当于人体的眼睛、耳朵、皮肤等感知器官，负责捕捉物理世界的各类信息，并将其转化为可传输、可处理的数字信号，是实现“万物互联”的前提和基础。

核心组件主要包括各类感知设备和被感知的实体对象。

(1) 感知设备：核心是各类传感器和识别设备，如电子标签(RFID)、读卡器、摄像头、红外感应器、人体感应器、温度传感器、湿度传感器、压力传感器等，这些设备能够精准捕捉物理世界的状态、变化和行为，例如人体感应器检测人员表征，摄像头捕捉图像信息，电子标签标识物品身份。

(2) 实体对象：即被感知、被连接的各类事物，涵盖人、服装、飞机、汽车、货架、工业设备、农业作物、家居用品等，感知设备通过与这些实体对象绑定，实现对其状态、位置、行为等信息的实时采集。

感知层的关键要求是“精准、高效、低功耗”，由于多数感知设备部署在户外或无人值守场景，需要在保证数据采集准确性的同时，降低能耗并提升稳定性，为后续的数据传输和处理提供可靠的原始数据。

2. 网络层

网络层位于感知层之上，其核心作用是“传输数据”，相当于人体的神经网络，将感

知层采集的数字信号，快速、安全、稳定地传输至上层的平台与云服务层，同时也负责将上层的控制指令(如远程控制信号)反向传输到感知层的设备中，实现双向数据交互。

核心组件是各类网络传输载体，可分为有线网络和无线网络两大类，覆盖不同场景的传输需求。

(1) 有线网络：适用于固定设备、高速传输、稳定性要求高的场景，主要包括拨号网络、局域网(LAN)、专有网络、专线网络等。例如工业场景中的设备通过专线网络传输大量生产数据，确保数据不丢失且低延迟。

(2) 无线网络：适用于移动设备、户外部署、不便布线的场景，主要包括2G/3G/4G等蜂窝网络、WLAN(无线局域网)、WiMAX等，通常用闪电符号表示无线传输。例如，户外摄像头通过4G网络传输图像、智能家居设备通过WLAN连接网络，实现灵活部署。

网络层的关键要求是“高速、稳定、安全、可扩展”，需要根据感知设备的数量、数据传输量、传输距离，选择合适的传输方式，同时防范数据传输过程中的泄露、篡改和丢失，确保数据能够精准送达目标节点。

3. 平台与云服务层

平台与云服务层是四层架构的核心枢纽，位于网络层与应用与用户交互层之间。其核心作用是“处理数据、管控设备、支撑应用”，相当于人体的大脑，承接网络层传输来的海量原始数据，进行清洗、存储、分析和处理，同时对感知层的设备进行统一管理，为上层应用提供标准化的服务和接口，是物联网系统实现智能化的核心支撑。

核心功能模块分为五大类，各模块协同工作，构建完整的平台支撑体系。

(1) 设备接入与管理：负责对接感知层的各类设备，实现设备的注册、登录、在线状态监控、远程调试、固件升级、设备注销等全生命周期管理，解决不同品牌、不同类型设备的兼容性问题，确保所有设备能够正常接入系统并稳定运行。

(2) 数据存储与处理：负责存储感知层采集的海量原始数据和处理后的数据，包括时序数据库(存储设备实时数据)、关系型数据库(存储设备信息、用户信息)等。同时，对数据进行清洗(剔除无效数据、异常数据)、转换(统一数据格式)、分析(挖掘数据价值，如异常预警、趋势分析)，将原始数据转化为有价值的信息。

(3) 消息队列与分发：负责处理系统内的各类消息交互，包括设备与平台的消息、平台与应用的消息、应用与用户的消息，实现消息的异步传输、排队处理和精准分发，避免消息拥堵，确保数据传输的及时性和可靠性。

(4) 应用开发与集成：提供标准化的API接口和开发工具，方便开发者快速开发各类物联网应用，无须关注底层的设备接入、数据传输和存储细节。同时，支持与第三方系统(如企业ERP系统、CRM系统)的集成，实现数据共享和业务协同。

(5) 安全与运维：负责整个物联网系统的安全防护和日常运维，包括设备安全(防止设备被入侵、篡改)、数据安全(防止数据泄露、篡改、丢失)、网络安全(防范网络攻击)。同时，提供系统监控、故障告警、日志分析等运维功能，确保系统长期稳定运行。

4. 应用与用户交互层

应用与用户交互层是物联网体系的最上层，其核心作用是“呈现价值、实现交互”，将平台与云服务层处理后的有价值信息通过各类载体呈现给用户，同时接收用户的操作指令，实现用户与物联网系统的双向交互，最终将物联网技术转化为具体的应用价值，服务

于各行各业。

核心组成分为应用载体和核心功能两大部分，贴合用户使用场景和行业需求。

(1) 应用载体：即用户与系统交互的具体渠道，覆盖不同用户的使用习惯，主要包括移动App(手机及平板端，方便用户随时随地操作)、Web端(电脑端，适合批量管理及详细数据分析)、小程序(轻量化载体，无须下载安装，便捷访问)、穿戴设备(如智能手表、手环，实现实时消息推送、简单操作)等。

(2) 核心功能：基于平台层提供的数据和服务，实现具体的应用场景功能。核心包括：数据可视化——通过图表、仪表盘等形式，直观呈现设备状态、数据变化趋势；远程控制——用户通过载体发送指令，远程控制感知层的设备，如远程开关灯、调节空调温度；报警通知——当设备出现异常或数据超出阈值时，通过短信、App推送等方式，及时提醒用户。此外，还可根据行业需求，延伸出工业监控、农业灌溉、健康医疗、公共安全、智能家居等个性化功能。

应用与用户交互层的关键要求是“便捷、直观、个性化”。需要根据不同行业、不同用户的需求，设计贴合场景的交互界面和功能模块，让用户能够快速上手，轻松实现对物联网系统的操作和管控，真正发挥物联网的应用价值。

5. 四层架构的协同逻辑

整个四层物联网体系架构的运行逻辑清晰且连贯：感知层采集实体对象的原始数据，并通过网络层传输至平台与云服务层；平台层对数据进行处理、分析，并对感知设备进行统一管理，生成有价值的信息和标准化服务；应用与用户交互层通过各类载体，将这些信息呈现给用户，同时接收用户指令并反向传递至感知层，实现“感知—传输—处理—交互”的闭环运行，最终达成“万物互联、智能管控”的目标。

2.1.1　感知层设备

感知层是物联网系统的数据源头，主要功能是实现物理世界信息的采集、识别与初步处理，通过各类终端设备将物理量、标识信息转化为可传输的数字信号。感知层设备涵盖传感器、RFID设备、嵌入式系统等，其性能与选型直接决定物联网系统的数据质量与可靠性。

2.1.2　各类传感器介绍

传感器作为感知层的主要感知单元，能将温度、湿度、光照、振动等非电物理量转化为电信号或数字信号，是物联网“感知能力”的主要部件。初级阶段需重点掌握以下几类主流传感器的特性、选型与应用场景。

1. 温湿度传感器

广泛应用于智能家居、仓储监控、农业大棚等场景，经典型号包括DHT11与DHT22。DHT11精度较低(温度±2℃、湿度±5%RH)，响应速度快、成本低，适合对精度要求不高的场景；DHT22精度更高(温度±0.5℃、湿度±2%RH)，测量范围更广(温度−40~80℃、湿度0~100%RH)，适用于工业监控、精密环境监测等场景。二者均采用单总线通信，接线简单，易于与嵌入式开发板对接。

常见的温湿度传感器如表2-1所示。

表 2-1　常见的温湿度传感器

传感器型号	接口	关键优势	主要缺点	典型应用场景
DHT11	单总线	价格极其便宜，易用	精度低，量程窄，速度慢	儿童教育、最低成本原型
DHT22	单总线	性价比高，精度够用	速度慢，偶尔读取出错	大部分DIY智能家居项目
AHT20	I2C	现代选择，精度好，接口标准	价格略高于DHT22	推荐用于大多数新项目
SHT3x	I2C	高精度高稳定，工业级品质	价格较高	科研、商用产品、高精度测量
BME280	I2C/SPI	集成气压，功能多，性能好	湿度精度略逊于SHT3x	气象站、需要气压或高度的项目

2. 光敏传感器

光敏传感器用于检测环境光照强度，经典型号为BH1750。该传感器采用I^2C通信协议，测量范围宽(1~65 535lx)，精度高(误差±20%)，可自动调节测量分辨率，广泛应用于智能照明系统、光伏设备监控、温室光照调控等场景，能根据光照强度自动触发设备启停(如路灯自动亮灭)。

常见的光敏传感器主要分为四种，其工作原理和适用场景各有不同，如表2-2所示。

表 2-2　常见的光敏传感器

传感器类型	核心特点	输出信号	典型应用场景	成本与复杂度
光敏电阻(LDR)	光线越强，电阻值越低。电路简单，需搭配分压电路使用	模拟量	环境光亮度检测、光照自动控制、光线触发开关	成本极低，入门首选
光电晶体管	响应速度极快，灵敏度高。常以模块形式出现，包含发射和接收管	数字开关量	物体检测(如计数、限位)、高速光电开关、精确测距	成本适中，应用专业
集成光强传感器模块	内部已集成信号处理电路，输出稳定的模拟电压，线性度好	模拟量	光强度定量测量、颜色识别、显示屏背光自动调节	成本较高，使用方便
紫外线(UV)传感器	专门用于检测紫外线波段(240~370nm)的光强度	模拟量	紫外线指数监测、防晒提醒、特定化学反应监测	特定用途，成本较高

3. 加速度传感器

加速度传感器广泛应用于工业设备监测、智能穿戴、运动检测等场景，经典型号包括ADXL345与MPU6050。ADXL345为纯三轴加速度检测，量程可调(±2g/±4g/±8g/±16g)，功耗极低，适合专注于振动与冲击监测的场景；MPU6050集成三轴加速度与三轴陀螺仪，支持姿态解算，可直接输出角度数据，适用于无人机、机器人等需要完整运动姿态感知的场景。二者均兼容I^2C通信，接线简洁，能快速与各类嵌入式开发板对接。

常见的加速度传感器如表2-3所示。

表 2-3　常见的加速度传感器

传感器型号	接口	关键优势	主要缺点	典型应用场景
ADXL345	I^2C/SPI	纯加速度专用，量程可调，抗冲击性好	无陀螺仪，需额外传感器进行姿态融合	工业振动监测、物流冲击检测、计步器

(续表)

传感器型号	接口	关键优势	主要缺点	典型应用场景
MPU6050	I^2C	六轴融合，自带DMP，可直接输出角度数据	温漂略大，长期静置精度会下降	无人机、平衡车、智能小车姿态控制
LIS3DH	I^2C/SPI	超小封装，低功耗，抗干扰强	输出数据率相对较低	可穿戴设备、医疗监测、微型传感器节点
MMA8452Q	I^2C	内置运动检测算法，中断功能丰富	量程较小(最大 ±8g)	手机翻转识别、笔记本电脑屏幕旋转、简单手势控制
BMA423	I^2C	博世品质，低功耗极致优化，寿命长	价格高于国产替代型号	高端可穿戴、物联网低功耗节点、电池供电设备

4. 其他常用传感器

除上述传感器外，还有广泛应用于安防监测、智慧农业、健康穿戴等场景的传感器，经典品类包括气体传感器、土壤湿度传感器、心率传感器等。MQ系列气体传感器可实现多类有害气体与烟雾检测，适配安防报警需求；FC-28土壤湿度传感器专为土壤墒情监测设计，适配智慧农业灌溉场景；MAX30102心率传感器集成血氧检测功能，是医疗健康穿戴设备的核心部件。各类传感器均可与嵌入式开发板简单对接，选型需结合场景需求综合考量。

其他常见的物联网传感器如表2-4所示。

表 2-4　其他常见的物联网传感器

传感器品类 / 型号	核心功能	关键优势	主要缺点	典型应用场景
气体传感器/MQ-2	检测烟雾、液化气、丙烷、酒精等	响应快，成本低，适配性广	精度一般，需定期校准	家庭安防报警、厨房燃气检测、火灾预警
土壤湿度传感器/FC-28	检测土壤含水量及墒情状态	接线简单，性价比高，适配各类土壤	长期浸泡易腐蚀，需做防水处理	智慧农业灌溉、盆栽智能浇水、大棚土壤监测
心率血氧传感器/MAX30102	检测心率、血氧饱和度及脉搏	低功耗，集成度高，检测精度高	受佩戴方式影响较大	智能手环/手表、医疗健康穿戴、运动监测设备
光照传感器/BH1750	检测环境光照强度(勒克斯)	数字输出，精度高，功耗极低	对强光直射适应性一般	智能家居灯光自动调节、温室光照监测、户外亮度采集
红外人体感应传感器/HC-SR501	检测人体移动及感应人体存在	感应距离可调，无需接触，易安装	易受温度及光线干扰，有检测盲区	楼道自动灯、安防人体感应报警、智能家电人体唤醒
超声波测距传感器/HC-SR04	检测障碍物距离及物体位置	测距范围广，精度尚可，成本低	受环境温度及障碍物材质影响	智能小车避障、垃圾桶满溢检测、停车场车位监测

2.1.3　RFID 技术原理与应用

RFID(Radio Frequency Identification，射频识别) 技术是一种非接触式自动识别技术，通过射频信号实现对物体的标识与数据读取，无须物理接触即可完成信息交互。相较于传统条形码，它具有读取距离远、可重复读写、抗干扰能力强及可批量识别等优势。

RFID系统由三部分组成：一是RFID标签(电子标签)，内置芯片与天线，存储物体唯一标识及相关信息，分为无源标签与有源标签。其中，无源标签无须供电，通过读写器射频信号获取能量，成本低、寿命长，但读取距离较近；有源标签自带电源，读取距离远、通信速度快，但成本高且寿命受电池限制。二是RFID读写器，负责发射射频信号并与标签交互数据，将标签信息转化为数字信号传输至嵌入式系统或云平台。三是天线，用于标签与读写器之间的射频信号传输，其形状与尺寸需根据读取距离、应用场景设计。

RFID技术的主要原理是电磁耦合：读写器通过天线发射特定频率的射频信号，标签进入信号覆盖范围后，无源标签通过电磁感应获取能量以激活芯片，有源标签主动响应信号，标签将存储的信息通过射频信号反馈给读写器，读写器对信号解码后完成数据采集。

RFID技术广泛应用于物流仓储(如货物全流程追溯)、门禁安防(如校园卡、企业工牌)、食品溯源(如生鲜产品产地追溯)、工业生产(如零部件流转管理)等场景，能够实现物体标识的自动化、规模化采集，从而提升管理效率。

2.1.4 嵌入式系统的角色

嵌入式系统是感知层的“核心控制器”，以嵌入式微处理器为核心，集成硬件(芯片、存储器、接口电路)与软件(操作系统、应用程序)，能够实现数据采集、处理、通信及设备控制等功能，是连接传感器、RFID设备与网络传输层的关键枢纽。在物联网系统中，嵌入式系统承担以下3大主要角色。

1. 数据采集与预处理中枢

嵌入式系统通过GPIO、I^2C、SPI、UART等接口连接各类传感器与RFID读写器，实时采集数据并进行初步预处理，包括过滤冗余数据、修正异常值、格式转换，将原始数据转化为符合通信协议的标准数据，减少后续网络传输与云端处理的压力。例如，通过嵌入式系统对DHT22采集的温湿度数据进行滑动平均滤波，可提升数据准确性。

2. 设备控制与联动核心

嵌入式系统可根据采集的数据或云端指令，控制执行器(如继电器、LED、电机)动作，实现物联网系统的闭环控制。例如，嵌入式系统检测到室内温度高于设定阈值时，自动控制继电器启动空调；接收云端“开灯”指令时，驱动LED点亮。

3. 网络接入与数据转发节点

嵌入式系统通过集成通信模组(如Wi-Fi、蓝牙、NB-IoT等)，将预处理后的数据上传至网络传输层，同时接收网络层下发的控制指令，实现感知层与网络层的数据交互。例如，基于ESP32的嵌入式系统，通过Wi-Fi模组将传感器数据上传至云平台，同时接收云平台的远程控制指令。

教学常用的嵌入式开发平台包括Arduino、STM32、ESP32等，其中Arduino门槛低、开源资源丰富，适合基础实操；STM32性能强、外设丰富，适用于工业级场景；ESP32集成Wi-Fi与蓝牙模组，性价比高，适合物联网无线通信场景。

2.1.5 实例：基于 Arduino 的温湿度采集系统

本实例将搭建一套基于Arduino Uno开发板与DHT11传感器的温湿度采集系统，实现

环境温湿度的实时采集与串口显示，为后续数据上传与设备控制奠定基础。

1. 硬件准备

主要器件：Arduino Uno开发板1块、DHT11温湿度传感器1个、10kΩ电阻1个、面包板1块、杜邦线若干。

硬件选型依据：Arduino Uno兼容性强、入门简单；DHT11成本低、接线简便，适合基础实操。

2. 硬件接线

(1) DHT11的VCC引脚接Arduino的5V引脚，GND引脚接Arduino的GND引脚(电源供电)。

(2) DHT11的DATA引脚接Arduino的数字引脚2，同时通过10kΩ电阻连接至VCC引脚(上拉电阻，用于保证数据传输稳定)。

(3) 面包板用于搭建临时电路，杜邦线实现各器件的引脚连接，确保接线牢固且无短路。

3. 软件开发

(1) 开发环境：Arduino IDE(可从官网https://www.arduino.cc/en/software/下载安装，支持Windows、Mac系统)。安装完成后，选择对应开发板(Tools→Board“Arduino UNO”→Arduino AVR Boards→Arduino UNO，如图2-2所示) 与端口(Tools→Port→对应COM口，如图2-3所示)。

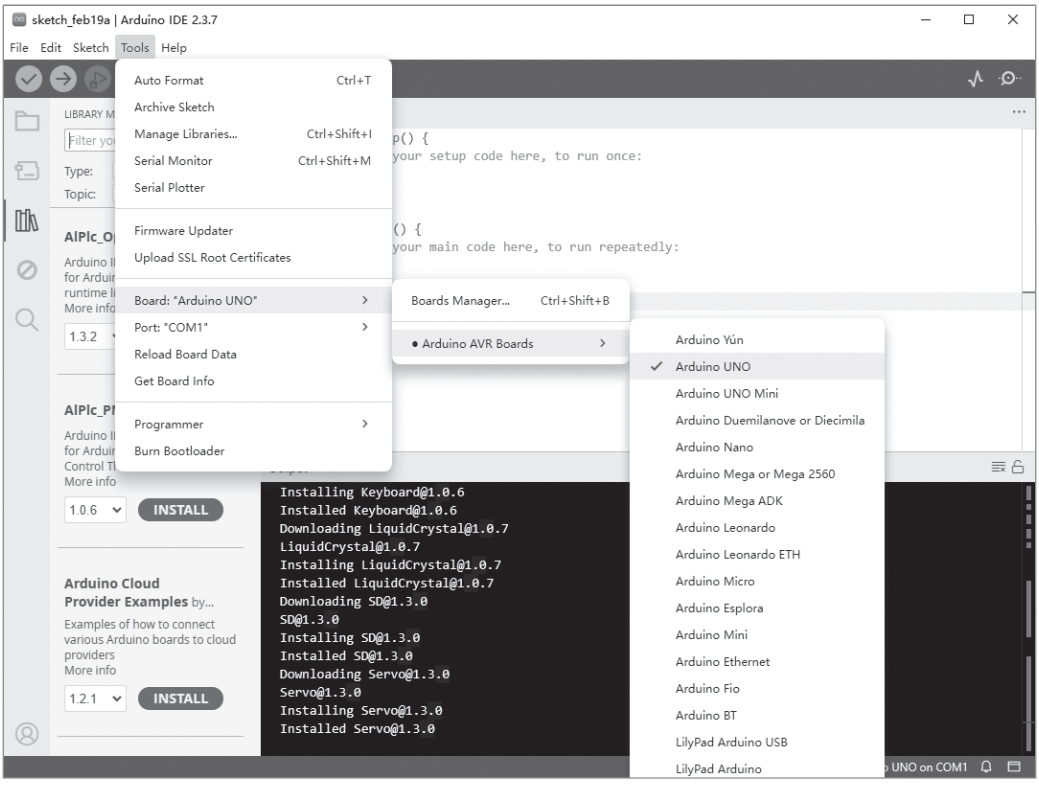

图 2-2　选择开发板

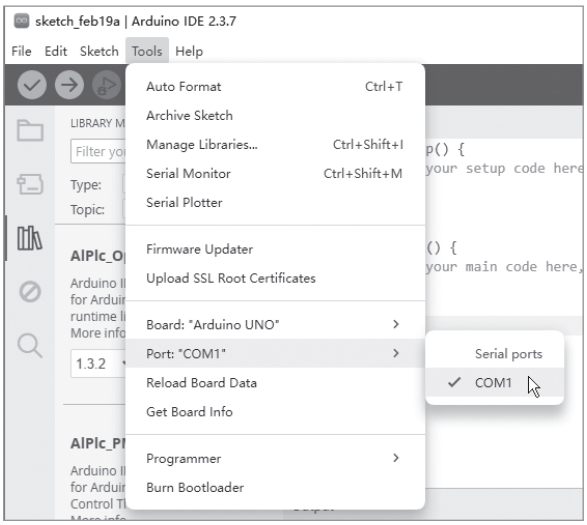

图 2-3　选择端口

(2) 库文件安装：DHT11需依赖第三方库。打开Arduino IDE，依次选择Sketch→Include Library→Manage Libraries，搜索DHT sensor library并安装，该库用于简化传感器数据读取的代码编写，如图2-4所示。

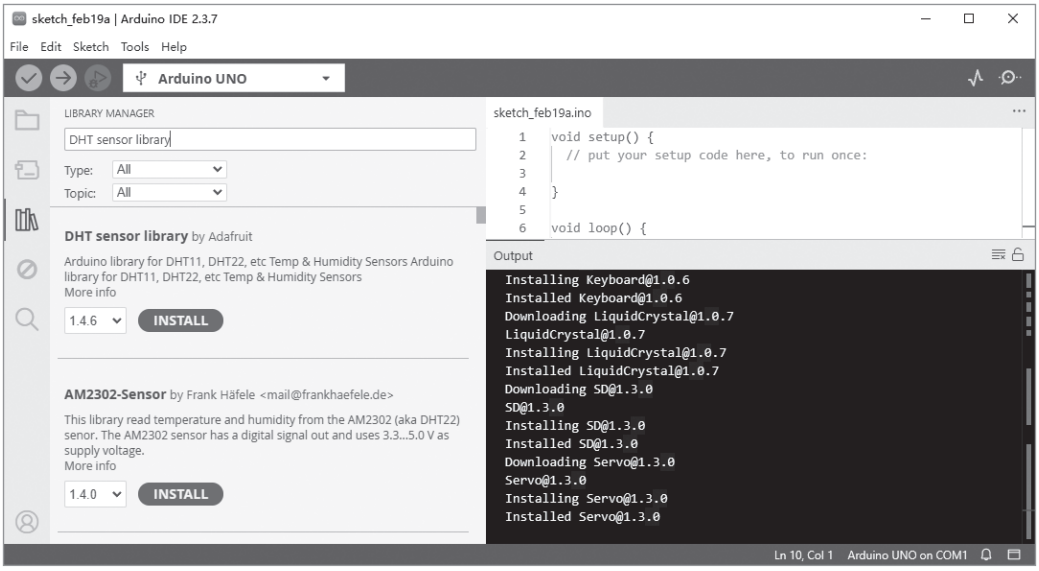

图 2-4　库文件安装

(3) 代码编写与上传：

```
//cpp
 // 引入DHT传感器库
 #include <DHT.h>
 // 定义DHT11连接引脚(Arduino数字引脚2)
 #define DHTPIN 2
 // 定义传感器类型为DHT11
 #define DHTTYPE DHT11
 // 初始化DHT对象
```

```
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  // 初始化串口通信，波特率9600(用于数据显示)
  Serial.begin(9600);
  // 初始化DHT传感器
  dht.begin();
  // 延时2秒，等待传感器稳定
  delay(2000);
}

void loop() {
  // 读取温湿度数据(DHT11读取间隔建议≥2秒，避免数据异常)
  delay(2000);
  float humidity = dht.readHumidity(); // 读取湿度
  float temperature = dht.readTemperature(); // 读取温度(摄氏度)

  // 校验数据是否有效
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("读取传感器数据失败，请检查接线！");
    return;
  }

  // 串口打印温湿度数据
  Serial.print("环境湿度：");
  Serial.print(humidity);
  Serial.print("%RH  ");
  Serial.print("环境温度：");
  Serial.print(temperature);
  Serial.println("℃");
}
```

代码说明：通过DHT库函数简化数据读取流程。setup()函数初始化串口与传感器，loop()函数周期性读取温湿度数据，校验数据有效性后通过串口输出，便于实时观察。

4. 运行程序

(1) 将编写好的代码上传至Arduino Uno开发板。上传成功后，打开串口监视器(Tools→Serial Monitor)，设置波特率为9600。

(2) 观察串口输出。若显示以下格式，说明系统运行正常：

```
环境湿度：45.0%RH  环境温度：28.0℃
环境湿度：45.0%RH  环境温度：28.0℃
环境湿度：46.0%RH  环境温度：28.0℃
环境湿度：45.0%RH  环境温度：29.0℃
```

若显示“读取传感器数据失败，请检查接线！”则需检查接线是否正确、传感器是否损坏及库文件是否安装到位：

```
读取传感器数据失败，请检查接线！
```

```
读取传感器数据失败，请检查接线！
读取传感器数据失败，请检查接线！
```

(3) 程序优化：更换DHT22传感器，修改代码中传感器类型为DHT22，对比两种传感器的精度差异；增加LED提示，当温度高于28℃时点亮LED，实现简单的阈值报警功能。

备注：关于Arduino开发技术与传感器数据采集技术，请参考相关文献。

2.2 网络传输层

网络传输层是物联网系统的“通信桥梁”，连接感知层与平台层，其核心功能是将感知层采集的数据可靠传输至云端平台，同时将云端的控制指令下发至感知层设备。该层级需根据感知层设备的分布、传输距离、功耗需求、带宽要求，选择合适的通信技术与协议，确保数据传输的高效性、稳定性与安全性。

2.2.1 有线通信与无线通信对比

物联网网络传输分为有线通信与无线通信两大类，二者在传输介质、性能指标、适用场景方面存在显著差异，选型需结合实际需求综合判断，对比如下。

(1) 传输介质与硬件成本：有线通信依赖网线、光纤、同轴电缆等物理介质，硬件成本(介质、交换机、路由器)随传输距离增加而上升，布线复杂，后期维护成本高；无线通信无须物理介质，依赖电磁波传输，硬件成本主要为通信模组与网关，布线灵活，适合设备分散、不易布线的场景(如户外环境监测、移动设备)。

(2) 性能指标：有线通信的带宽高(如千兆网线带宽可达1000Mbps)，传输延迟低至毫秒级，抗干扰能力强，不受电磁辐射影响，数据传输稳定性高；无线通信受频段、距离、干扰影响，带宽与延迟差异较大(如Wi-Fi带宽可达数百Mbps，NB-IoT带宽仅几十Kbps)，抗干扰能力较弱，但具备移动性与广覆盖优势。

(3) 功耗特性：有线通信设备可通过市电供电，无功耗限制，适合固定设备；无线通信设备多依赖电池供电，低功耗技术(如LoRa、NB-IoT)可延长续航时间，高功耗技术(如Wi-Fi)续航时间较短，需权衡功耗与性能。

(4) 适用场景：有线通信适用于固定设备及对带宽与稳定性要求高的场景(如工业厂房设备互联、数据中心与云平台通信)；无线通信适用于设备分散、移动性强、布线困难的场景(如智能家居、智慧城市、户外农业监测)。

实际物联网系统中，常采用“有线+无线”混合通信模式：近距离设备通过无线通信汇聚数据至网关，网关再通过有线通信上传至云端，兼顾灵活性与稳定性。

2.2.2 Wi-Fi、蓝牙、ZigBee、LoRa、NB-IoT 简介

无线通信是物联网网络传输层的核心，需要重点掌握五种主流短距离与低功耗广域网技术，其特性、优势与适用场景如下。

(1) Wi-Fi：短距离无线通信技术，工作频段为2.4GHz/5GHz，带宽可达150Mbps~1Gbps，传输距离10~100米，支持TCP/IP协议，可直接接入互联网。优势：带宽高、接入便捷、与互联网兼容性强；不足：功耗较高、抗干扰能力一般(2.4GHz频段设备密集)。适用于智能家居(如智能电视、空调)、室内监控、需要高速传输数据的场景。

(2) 蓝牙：短距离无线通信技术，工作频段为2.4GHz，带宽1~3Mbps，传输距离1~10米，功耗低于Wi-Fi。蓝牙4.0及以上版本支持BLE(蓝牙低功耗) 模式，功耗大幅降低，续航可达数月。优势：低功耗、成本低、配对快速；不足：传输距离短、连接设备数量有限(最多8个)。适用于智能穿戴设备(手环、手表)、近距离设备联动(如手机控制智能灯)、低速率数据传输等场景。

(3) ZigBee：短距离低功耗无线通信技术，工作频段2.4GHz，带宽250Kbps，传输距离10~100米，支持多节点组网(最多65 535个节点)，功耗低、抗干扰能力强。优势：组网能力强、低功耗、可靠性高；不足：带宽低、传输距离有限。适用于智能楼宇(灯光、门禁联动)、工业控制、传感器网络(多节点数据汇聚)等场景。

(4) LoRa：低功耗广域网技术，工作在非授权频段(如433MHz)，带宽0.3~50Kbps，传输距离1~15公里，功耗极低，电池续航可达数年，支持多节点组网。优势：广覆盖、低功耗、抗干扰能力强；不足：带宽低、传输延迟高。适用于户外环境监测(农业、环保)、智能抄表(水电燃气表)、远距离低速率数据传输等场景。

(5) NB-IoT：基于蜂窝网络的低功耗广域网技术，工作在授权频段，带宽160~250Kbps，传输距离数公里至数十公里，功耗低，电池续航可达数年，由运营商部署网络，覆盖范围广。优势：覆盖广、稳定性高、无须自建网关；不足：带宽较低、需缴纳运营商资费。适用于智慧城市(智能井盖、路灯)、智能抄表、远程设备监控(无自建网络条件的场景)等场景。

(6) 技术选型原则：短距离、高带宽选Wi-Fi；短距离、低功耗、近距离联动选蓝牙BLE；短距离、多节点组网选ZigBee；长距离、低功耗、非授权频段选LoRa；长距离、低功耗、需广覆盖选NB-IoT。

2.2.3　蜂窝网络在物联网中的应用

蜂窝网络(2G/3G/4G/5G)是基于基站组网的无线通信网络，由运营商部署，具备广覆盖、移动性强、稳定性高的优势。在物联网中主要用于长距离及移动性场景的数据传输，是无线通信技术的重要补充。

(1) 各代蜂窝网络的物联网应用场景如下。

- 2G/3G：技术成熟、覆盖广，但带宽低、功耗较高，目前主要用于传统智能抄表、简单设备远程监控(如老旧电梯监控)，正逐步被4G/5G替代。
- 4G：带宽高(下行速率可达100Mbps)、延迟低(约50ms)、覆盖完善，适用于车载物联网(如网约车定位、车载娱乐)、远程视频监控(如道路监控)、移动设备数据传输(如快递柜)等场景。
- 5G：具备高带宽(下行速率达10Gbps)、低延迟(1ms以内)、广连接(每平方公里百万级连接)三大特性，适用于工业物联网(远程设备控制、机器视觉质检)、车联网(自动驾驶协同)、高清视频监控(如智慧交通)、VR/AR物联网应用等复杂场景。

(2) 蜂窝网络物联网设备的核心组件为：通信模组与SIM卡。通信模组集成蜂窝网络通信芯片，实现设备与基站的信号交互，常用模组有4G Cat.1模组(性价比高，适用于中低速场景)及5G模组(高性能，适用于复杂场景)。SIM卡分为普通SIM卡与eSIM卡(嵌入式SIM，无须物理卡槽，可远程写号及切换运营商)，适合小型设备与批量部署场景。

(3) 蜂窝网络在物联网中的优势与局限：优势在于广覆盖(无须自建网络，偏远地区也可接入)、移动性强(支持设备动态移动过程中通信)、稳定性高(运营商专业运维)。局限

在于成本较高(模组费用及资费套餐)、功耗高于LoRa/NB-IoT(不适用于电池供电的长待机设备)。

2.2.4 实例：使用 ESP32 连接 Wi-Fi 上传数据

本实例基于ESP32开发板(集成Wi-Fi模组)，将DHT11采集的温湿度数据通过Wi-Fi上传至免费物联网数据可视化平台(Thingspeak)，实现数据的远程查看，从而掌握感知层与网络层的数据交互流程。

1. 硬件准备

核心器件：ESP32开发板1块、DHT11传感器1个、10kΩ电阻1个、面包板1块、杜邦线若干。

ESP32优势：集成Wi-Fi与蓝牙模组，性价比高，无须额外添加通信模组，适合物联网无线传输场景。

2. 硬件接线

(1) DHT11的VCC接ESP32的3.3V引脚(ESP32支持3.3V供电，避免5V烧毁器件)，GND接ESP32的GND引脚。

(2) DHT11的DATA引脚接ESP32的GPIO2引脚，同时通过10kΩ电阻接3.3V引脚(上拉电阻)。

(3) 确保接线无短路，ESP32通过USB线连接电脑(供电及数据传输)。

3. 平台准备(Thingspeak平台配置)

(1) 访问Thingspeak官网(https://thingspeak.com/)，注册账号并登录，创建新频道(Channels→New Channel)。

(2) 开启两个字段(Field1设为Temperature，Field2设为Humidity)，填写频道名称(如ESP32温湿度监测)，保存频道，如图2-5所示。

图 2-5 开启两个字段

(3) 记录频道API Key(API Keys→Write API Key)，该密钥用于ESP32向平台上传数据的身份认证；同时记录频道ID，用于查看数据，如图2-6所示。

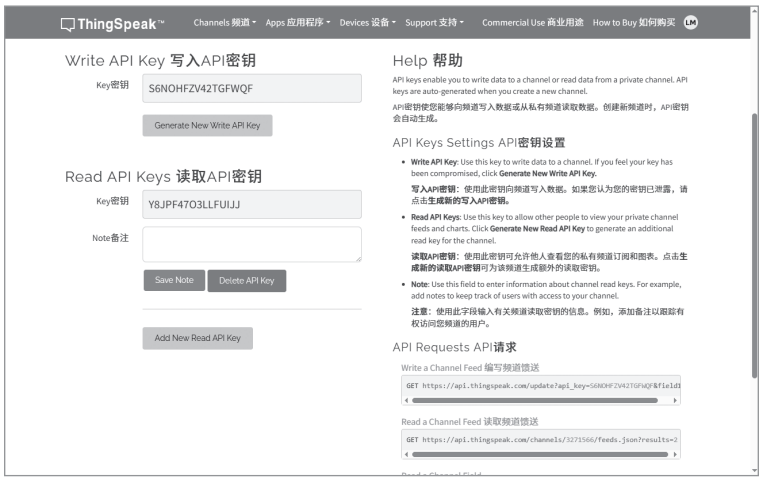

图 2-6　查看 API Keys

4. 软件开发

(1) 开发环境：Arduino IDE，安装ESP32开发板支持包(文件→首选项→附加开发板管理器网址，添加https://dl.espressif.com/dl/package_esp32_index.json，然后通过开发板管理器安装“esp32”包)；安装DHT传感器库(前面已介绍)。

(2) 代码编写与上传。本示例使用的是Thingspeak的仿真环境，因此需替换代码中的Wi-Fi名称、密码及Write API Key。如果使用的是Arduino IDE环境，直接替换为真实Wi-Fi即可。

```
#include <DHT.h>
#include <WiFi.h>
#include <HTTPClient.h>

// ========================= 仿真配置注意 =========================
// 1. Wokwi仿真中Wi-Fi名称/密码可填任意值(无须真实Wi-Fi)
// 2. API Key需替换为自己的Thingspeak Write API Key
// ================================================================
const char* ssid = "Wokwi-GUEST";      // 仿真Wi-Fi名称(固定填Wokwi-Fi)
const char* password = "";             // 仿真Wi-Fi密码(留空即可)
const char* apiKey = "S6NOHFZV42TGFWQF"; // 替换为自己的Write API Key
const char* server = "api.thingspeak.com";
String url = "http://" + String(server) + "/update?api_key=" + String(apiKey) +
                  "&field1=" + String(temperature) + "&field2=" + String(humidity);

// DHT11配置(仿真中同样使用GPIO2，与实体接线一致)
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

WiFiClient client;

void setup() {
  Serial.begin(115200);
```

```
    dht.begin();
    delay(2000);

    // 连接仿真Wi-Fi
    Serial.print("正在连接Wi-Fi：");
    Serial.println(ssid);
    WiFi.begin(ssid, password);

    // 仿真中Wi-Fi连接会瞬间完成，保留等待逻辑保证兼容性
    int wifiRetry = 0;
    while (WiFi.status() != WL_CONNECTED && wifiRetry < 20) {
      delay(500);
      Serial.print(".");
      wifiRetry++;
    }

    if (WiFi.status() == WL_CONNECTED) {
      Serial.println("\nWi-Fi连接成功，IP地址：");
      Serial.println(WiFi.localIP());
    } else {
      Serial.println("\nWi-Fi连接失败(仿真环境请检查配置)！");
    }
}

void loop() {
    // 步骤1：读取温湿度数据(仿真中DHT11会生成模拟数据)
    delay(2000);
    float humidity = dht.readHumidity();
    float temperature = dht.readTemperature();

    // 数据校验
    if (isnan(humidity) || isnan(temperature)) {
      Serial.println("读取传感器数据失败！");
      delay(1000);
      return;
    }

    // 步骤2：上传数据到Thingspeak
    if (WiFi.status() == WL_CONNECTED) {
      HTTPClient http;
      String url = "http://" + String(server) + "/update?api_key=" + String(apiKey) +
                   "&field1=" + String(temperature) + "&field2=" + String(humidity);

      http.begin(client, url);
      int httpCode = http.GET();
      String payload = http.getString();

      // 打印调试信息
      Serial.print("HTTP响应码：");
      Serial.println(httpCode); // 200=成功，400=API Key错误，429=频率超限
      Serial.print("温湿度数据：");
```

```
        Serial.print(temperature);
        Serial.print("℃, ");
        Serial.print(humidity);
        Serial.println("%RH");

        http.end();
    } else {
        Serial.println("Wi-Fi断开，尝试重连...");
        WiFi.begin(ssid, password);
        delay(1000);
    }

    // Thingspeak免费版15秒限制(仿真中也需遵守)
    delay(15000);
}
```

以上代码通过WiFi库实现ESP32与Wi-Fi的连接，HTTPClient库发送HTTP GET请求向Thingspeak上传数据，包含Wi-Fi重连机制与数据校验，确保传输稳定性。

其中，DHT.h、WiFi.h和HTTPclient.h三个库是实现功能的基础，必须确保正确安装(DHT库需手动安装，WiFi及HTTPClient是ESP32核心库，安装开发板支持包后自带)。

代码中的关键替换项如下。

(1) ssid/password：改为你实际的Wi-Fi名称(注意区分大小写)和密码。

(2) apiKey：改为Thingspeak账号中对应频道的Write API Key(在频道→API Keys 中查看) 。

(3) 硬件关联：DHTPIN 2表示DHT11的数据引脚接ESP32的GPIO2，若接了其他引脚(如GPIO4)，需同步修改此处。

5.运行程序

(1) 上传代码至ESP32，打开串口监视器(波特率115200)，观察是否显示“Wi-Fi连接成功”与温湿度数据：

```
正在连接Wi-Fi：Wokwi-GUEST
...
Wi-Fi连接成功，IP地址：
192.168.4.1
HTTP响应码：200
温湿度数据：27.8℃, 46.2%RH
HTTP响应码：200
温湿度数据：28.1℃, 45.9%RH
HTTP响应码：200
温湿度数据：27.9℃, 46.0%RH
...
```

(2) 访问Thingspeak频道页面(通过频道ID)，查看实时数据与趋势图表，若能正常显示温湿度变化曲线，说明数据上传成功。

(3) 实操拓展：添加LED报警功能，当温度高于30℃时，ESP32控制LED点亮；通过Thingspeak平台设置阈值报警，当数据超出设定范围时发送邮件通知。

2.3 平台与云服务层

平台与云服务层是物联网系统的“中枢大脑”，连接网络传输层与应用交互层。核心功能是实现设备管理、数据存储、消息分发、业务逻辑处理等，为应用层提供标准化的接口与服务，从而降低物联网应用开发的门槛。该层级集成云计算、大数据、数据库等技术，实现物联网系统的规模化管控与智能化分析。

2.3.1 云平台功能与分类

物联网云平台是整合硬件接入、数据处理、应用开发的核心载体，具备全链路的物联网服务能力，其核心功能与分类如下。

1. 核心功能

(1) 设备接入与管理：提供标准化的设备接入协议如MQTT、HTTP、CoAP，支持海量设备的同时接入；实现设备全生命周期管理，包括设备注册、身份认证、在线状态监测、远程控制、固件OTA升级及设备分组管理等，确保设备接入的安全性与可控性。

(2) 数据存储与处理：提供海量物联网数据的存储服务，支持时序数据库(存储传感器实时数据)、关系型数据库(存储设备信息、用户数据)等多种存储方式；对上传的数据进行清洗、过滤、聚合等预处理，支持实时数据分析与离线数据挖掘，提取有价值的信息。

(3) 消息队列与分发：通过消息队列机制如MQTT消息队列实现设备与平台、设备与设备之间的异步通信，解耦设备与应用，避免数据拥堵，提升系统稳定性；支持消息的定向分发、广播分发，满足不同场景的通信需求。

(4) 应用开发与集成：提供API接口、SDK开发工具包、可视化开发平台，支持开发者快速搭建物联网应用，无须关注底层硬件接入与数据处理；支持与第三方系统集成，如企业ERP系统、微信公众号，拓展应用边界。

(5) 安全与运维：提供设备身份认证、数据加密传输、访问权限控制等安全机制，保障设备与数据安全；提供系统监控、日志审计、故障告警等运维功能，便于开发者排查问题并优化系统。

2. 分类

(1) 按服务类型分类如下。

- IaaS(基础设施即服务)：提供底层计算、存储、网络资源，如阿里云ECS、AWS EC2，适用于需要自定义开发物联网平台的企业。
- PaaS(平台即服务)：提供标准化的物联网平台服务，包括设备接入、数据处理、API接口等，如阿里云IoT Core、华为云IoT，是本科阶段实操与企业应用的主流方案。
- SaaS(软件即服务)：提供现成的物联网应用解决方案，如智能家居管理平台、工业监控系统，用户无须开发，直接使用服务。

(2) 按部署方式分类如下。

- 公有云平台：由第三方服务商部署与运维，资源共享，成本低、扩展性强，适合中小企业与个人开发者，如阿里云、AWS IoT。
- 私有云平台：部署在企业内部网络，数据与资源私有化，安全性高且可控性强，

适合对数据安全要求高的大型企业与工业场景，如华为云私有IoT平台。

- 混合云平台：结合公有云与私有云的优势，核心数据存储在私有云，非核心业务使用公有云资源，兼顾安全性与扩展性。

2.3.2　AWS IoT Core、阿里云 IoT、华为云 IoT 介绍

目前全球主流的物联网PaaS云平台包括AWS IoT Core(国外)、阿里云IoT Core、华为云IoT(国内)，三者各有优势，适配不同的开发需求与场景，核心特性如下。

(1) AWS IoT Core：亚马逊旗下的全球物联网云平台，具备全球化部署优势，支持多区域、多终端接入。主要特性包括：支持MQTT、HTTP、WebSocket等多种通信协议，支持海量设备接入与管理；集成AWS Lambda(无服务器计算)、Amazon S3(存储)、Amazon AI(人工智能)等服务，可实现数据的深度分析与智能决策；提供完善的安全机制，包括设备身份认证、数据加密、访问控制等。适合需要全球化部署的物联网应用，但其国内访问速度较慢，学习成本较高。

(2) 阿里云IoT Core：阿里云旗下物联网平台，国内市场占有率高，生态完善，入门门槛低。主要特性包括：支持MQTT、CoAP、HTTP等协议，提供设备接入、管理、OTA升级等全链路服务；集成阿里云时序数据库(InfluxDB)、大数据分析(MaxCompute)、可视化工具(DataV) 等服务，实现数据存储与可视化一体化；提供丰富的SDK与Demo案例，支持Arduino、ESP32、STM32等多种开发板快速接入。适合国内开发者、中小企业与本科阶段实操，提供免费额度，可降低开发成本。

(3) 华为云IoT：华为旗下物联网平台，依托华为的通信技术优势，在工业物联网、5G物联网领域表现突出。主要特性包括：支持海量设备接入，具备低延迟、高可靠的通信能力，适配工业级场景需求；集成华为云AI、数字孪生、边缘计算等服务，支持AIoT与工业互联网解决方案；提供完善的安全体系，符合国内数据安全法规；支持华为自研芯片与设备的深度适配。适合工业物联网、智慧城市等大型项目，同时提供免费额度，适合学习与原型开发。

本科阶段实操推荐选择阿里云IoT Core或华为云IoT，其国内访问稳定、文档丰富、入门简单，能够快速完成设备接入与应用开发。

2.3.3　设备管理与消息队列机制

设备管理与消息队列是物联网云平台的核心机制，直接影响系统的稳定性、安全性与可扩展性。其具体原理与应用如下。

1. 设备管理

设备管理涵盖设备从注册到退市的全生命周期，核心流程与功能包括以下内容。

(1) 设备注册与身份认证：设备接入云平台前需完成注册，平台分配唯一的设备标识(Device ID) 与身份凭证(如三元组：ProductKey、DeviceName、DeviceSecret)。设备接入时，通过身份凭证与平台进行双向认证，确保接入设备的合法性，防止非法设备入侵。例如，阿里云IoT采用三元组认证，设备发送认证请求时，平台验证凭证有效性，通过后方可接入。

(2) 设备状态监测与管控：平台实时监测设备的在线/离线状态、信号强度、数据上传频率等信息，并通过可视化界面展示；支持远程控制设备，下发控制指令(如启动、停止、参数调整)；支持设备分组管理，对批量设备进行统一管控(如批量升级固件)，从而提升管

理效率。

(3) 固件OTA升级：通过云平台远程向设备推送固件更新包，设备接收后自动升级，无须人工现场操作，适合大规模设备部署场景。升级过程中需具备断点续传、版本回滚机制，确保升级失败后设备可恢复正常运行，避免设备“变砖”。

2. 消息队列机制

物联网系统中，设备与平台、设备与设备之间的消息交互具有异步性、高并发、多节点的特点，消息队列机制可有效解决这些问题，其核心原理与优势如下。

(1) 核心原理：消息队列是一个临时存储消息的缓冲区，发送方(设备或平台)将消息发送至队列后即可返回，无须等待接收方响应；接收方(设备或平台)从队列中异步读取消息并处理，从而实现发送方与接收方的解耦。物联网中常用的消息队列协议为MQTT(Message Queuing Telemetry Transport)。该协议轻量、低功耗且适合低带宽场景，是设备与云平台通信的主流协议。

(2) 核心优势如下。

- 削峰填谷：应对高并发消息(如海量设备同时上传数据)，队列缓存消息，避免平台服务器过载。
- 解耦系统：发送方与接收方无须直接通信，只需约定消息格式，便于系统扩展与维护。
- 异步通信：支持设备离线时缓存消息，设备上线后自动接收消息，确保消息不丢失。
- 灵活路由：支持消息按主题(Topic)路由，实现定向分发(如指定设备接收指令)、广播分发(如所有设备接收通知)。

例如，在智能家居系统中，手机App发送“开灯”指令至云平台消息队列，智能灯设备订阅对应主题，从队列中读取指令后执行开灯操作。App无须等待灯的响应，从而提升用户体验。

2.3.4 实例：将设备接入阿里云 IoT 平台

本实例基于ESP32与DHT11，实现设备接入阿里云IoT Core平台，完成温湿度数据上传与云端远程控制LED功能，从而掌握设备接入云平台的核心流程与消息交互机制。

1. 阿里云IoT平台准备

(1) 注册并登录阿里云账号，进入IoT Core控制台(https://iot.console.aliyun.com/)。创建产品(产品→创建产品，选择“自定义品类”，通信方式选择Wi-Fi，数据格式选择JSON)，记录产品的ProductKey。

(2) 在产品下添加设备(设备→添加设备)，填写设备名称，创建后获取设备的DeviceName与DeviceSecret(三元组：ProductKey、DeviceName、DeviceSecret，用于设备认证)。

(3) 创建Topic(通信主题)。选择产品→Topic类列表→自定义Topic，创建以下两个Topic。

① 数据上传Topic：/a1XXXXXXXXX/${deviceName}/upload(设备向平台上传温湿度数据)。

② 指令下发Topic：/a1XXXXXXXXX/${deviceName}/control(平台向设备下发LED控制指令)。

(4) 配置规则引擎(可选)。将上传的数据转发至阿里云时序数据库或可视化工具，便于数据存储与查看。

2. 硬件准备与接线

(1) 硬件：ESP32开发板、DHT11传感器、10kΩ电阻、LED灯、220Ω限流电阻、面包板、杜邦线。

(2) 接线：DHT11接线同2.2.4节的实例；LED正极通过220Ω电阻接ESP32 GPIO4引脚，负极接GND引脚。

3. 软件开发(基于Arduino IDE)

(1) 安装阿里云IoT SDK：打开Arduino IDE，通过库管理器安装AliyunIoTSDK库，用于简化设备接入与消息交互代码。

(2) 代码编写与上传(替换三元组及Wi-Fi信息)：

```
//cpp
 #include <DHT.h>
 #include <WiFi.h>
 #include <AliyunIoTSDK.h>

 // Wi-Fi配置
 const char* ssid = "YOUR_WIFI_NAME";
 const char* password = "YOUR_WIFI_PASSWORD";

 // 阿里云IoT三元组(替换为自己的信息)
 #define PRODUCT_KEY "YOUR_PRODUCT_KEY"
 #define DEVICE_NAME "YOUR_DEVICE_NAME"
 #define DEVICE_SECRET "YOUR_DEVICE_SECRET"

 // DHT11配置(GPIO2)
 #define DHTPIN 2
 #define DHTTYPE DHT11
 DHT dht(DHTPIN, DHTTYPE);

 // LED配置(GPIO4)
 #define LED_PIN 4

 // Wi-Fi连接函数
 void connectWiFi() {
    Serial.print("连接Wi-Fi：");
    Serial.println(ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
       delay(500);
       Serial.print(".");
    }
    Serial.println("\nWi-Fi连接成功！");
 }

 // 阿里云消息回调函数(处理平台下发的控制指令)
```

```
void onMessageReceived(char* topic, byte* payload, unsigned int length) {
  Serial.print("收到指令：");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();

  // 解析JSON格式指令({"led":"on"}或{"led":"off"})
  DynamicJsonDocument doc(1024);
  deserializeJson(doc, payload, length);
  const char* ledStatus = doc["led"];
  if (strcmp(ledStatus, "on") == 0) {
    digitalWrite(LED_PIN, HIGH); // 点亮LED
    Serial.println("LED已点亮");
  } else if (strcmp(ledStatus, "off") == 0) {
    digitalWrite(LED_PIN, LOW); // 熄灭LED
    Serial.println("LED已熄灭");
  }
}

void setup() {
  Serial.begin(115200);
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW);
  dht.begin();
  delay(2000);

  connectWiFi(); // 连接Wi-Fi

  // 初始化阿里云IoT连接
  AliyunIoTSDK::begin(Serial, PRODUCT_KEY, DEVICE_NAME, DEVICE_SECRET);
  // 订阅指令下发Topic
  AliyunIoTSDK::subscribe("/" PRODUCT_KEY "/" DEVICE_NAME "/control");
  // 设置消息回调函数
  AliyunIoTSDK::setCallback(onMessageReceived);
}

void loop() {
  AliyunIoTSDK::loop(); // 处理阿里云IoT消息

  // 读取温湿度数据(每5秒上传一次)
  static unsigned long lastUploadTime = 0;
  if (millis() - lastUploadTime > 5000) {
    float humidity = dht.readHumidity();
    float temperature = dht.readTemperature();

    if (!isnan(humidity) && !isnan(temperature)) {
      // 构建JSON格式数据
      DynamicJsonDocument doc(1024);
      doc["temperature"] = temperature;
      doc["humidity"] = humidity;
```

```
            String json;
            serializeJson(doc, json);

            // 上传数据至阿里云IoT平台
            AliyunIoTSDK::publish("/" PRODUCT_KEY "/" DEVICE_NAME "/upload", json.c_str());
            Serial.print("上传数据：");
            Serial.println(json);

            lastUploadTime = millis();
          } else {
            Serial.println("读取传感器数据失败！");
          }
        }
    }
```

代码说明：通过AliyunIoTSDK实现设备与阿里云的连接、认证与消息交互；回调函数处理云端下发的LED控制指令，解析JSON格式数据；周期性上传温湿度数据至云端。

4. 运行程序

(1) 上传代码至ESP32，打开串口监视器(波特率 115200)，观察是否显示“Wi-Fi连接成功！”及“[AliyunIoTSDK] Connected successfully!”：

```
连接Wi-Fi：YOUR_WIFI_NAME
...
Wi-Fi连接成功！
[AliyunIoTSDK] Connecting to Aliyun IoT Platform...
[AliyunIoTSDK] Connected successfully!
[AliyunIoTSDK] Subscribed to: /YOUR_PRODUCT_KEY/YOUR_DEVICE_NAME/control
上传数据：{"temperature":28.0,"humidity":45.0}
上传数据：{"temperature":28.2,"humidity":44.8}
// 收到平台下发指令 {"led":"on"} 时：
收到指令：{"led":"on"}
LED已点亮
上传数据：{"temperature":28.1,"humidity":45.0}
// 收到平台下发指令 {"led":"off"} 时：
收到指令：{"led":"off"}
LED已熄灭
上传数据：{"temperature":27.9,"humidity":45.2}
...
```

(2) 数据上传测试。在阿里云IoT控制台(设备→监控运维→设备日志)中，查看设备上传的温湿度数据，确认数据上传正常。

(3) 远程控制测试。在控制台(设备→在线调试)中，选择指令下发Topic，发送JSON格式指令({"led":"on"})，观察LED是否点亮；发送{"led":"off"}，观察LED是否熄灭，同时查看串口监视器的指令接收日志。

2.4 应用与用户交互层

应用与用户交互层是物联网系统的“价值出口”，直接面向用户与具体业务场景。其

核心功能是将平台与云服务层处理后的数据转化为用户可感知、可操作的服务。该层级以用户需求为导向，通过移动App、Web端、小程序等载体，实现数据可视化、远程控制、报警通知等功能，同时借助用户行为数据分析优化服务体验，构建“数据—服务—用户”的闭环，是物联网系统实现商业价值与用户价值的关键环节。

2.4.1 移动 App 与 Web 端展示

移动App与Web端是应用与用户交互层的两大核心载体，二者功能互补、场景适配，共同覆盖个人用户与企业用户的多样化需求。其技术选型、核心优势与适用场景需结合物联网特性精准定位。

1. 移动App展示

移动App以智能手机及平板为终端，具备便携性、实时性、个性化的优势，适合个人用户随时随地实现远程控制与数据查看，是物联网消费级场景的主流交互载体。

从技术选型来看，教学常用的开发框架分为原生开发与跨平台开发两类。原生开发(iOS端Swift/Objective-C、Android端Java/Kotlin)性能稳定、适配性强，能深度调用设备硬件(摄像头、蓝牙)，适合对体验要求较高的场景，但需开发两套代码，效率较低；跨平台开发(Flutter、React Native)采用一套代码适配双端，开发效率高、迭代速度快，开源资源丰富，是教学实操与中小企业开发的首选。其中Flutter基于Dart语言，性能接近原生，界面一致性强，在物联网App开发中应用广泛。

移动App的核心功能模块包括：设备列表管理(添加、删除及分组设备，查看设备在线状态)、实时数据展示(温湿度、光照等参数直观呈现)、远程控制(下发开关、参数调节指令)、历史数据查询、个性化场景设置(如智能家居的“起床模式”“睡眠模式”)、消息通知(报警提醒、设备状态变更通知)。例如，智能家居App可让用户在公司远程查看室内温湿度，一键开启空调；智能穿戴App可实时展示心率及步数数据，生成健康报告。

2. Web端展示

Web端以电脑及大屏显示器为终端，具备展示空间大、数据承载能力强、多用户共享的优势，适合企业级监控、公共服务场景，如工业控制中心、智慧城市运营平台、仓储监控系统等。

技术选型上，教学中需重点掌握前端开发框架与数据可视化工具的结合。前端框架常用Vue.js及React，二者均为开源框架，组件化开发模式便于复用与维护。其中Vue.js入门门槛较低、文档丰富，更适合初学者；数据可视化工具可搭配ECharts、Chart.js，支持折线图、柱状图、仪表盘、热力图等多种图表类型，能满足物联网场景下实时数据、趋势数据、多维度数据的展示需求。此外，Web端还可结合Node.js、SpringBoot等后端框架搭建服务，实现与物联网云平台的接口对接，完成数据获取与指令下发。

Web端的核心功能模块包括：多设备集中监控(同时展示海量设备的在线状态、关键参数，支持设备分组查看)、数据可视化报表(生成日/周/月历史数据报表，支持数据导出与打印，为决策提供依据)、远程控制面板(针对不同设备提供标准化控制界面，如灯光亮度调节、空调温度设置)、系统权限管理(分配管理员、操作员、普通用户等不同角色，设置数据查看、设备控制等权限，保障系统安全)、运维日志记录(留存设备操作记录、数据异常日志，便于故障追溯与系统优化)。例如，工业监控Web端可在大屏上实时展示车间所有

设备的运行参数，当某设备振动值超标时，界面自动标红提醒，管理员单击设备图标即可下发停机指令。

移动App与Web端的协同逻辑：二者共享云平台数据接口，实现数据同步与功能互补。个人用户通过移动App实现碎片化场景的快速操作，企业用户通过Web端完成规模化设备管控与数据复盘。例如，智能家居用户用App睡前关闭设备，物业管理员通过Web端统计整栋楼设备能耗数据。

2.4.2　用户行为数据分析

用户行为数据分析是物联网应用层优化的核心手段，通过采集、挖掘用户与设备的交互数据，精准识别用户需求、优化产品功能并提升服务质量，同时为业务决策提供数据支撑。其核心价值在于将“用户操作”转化为“可量化数据”，打破“经验驱动”的优化模式，实现“数据驱动”的迭代升级。

1. 核心数据采集范围

物联网场景下的用户行为数据需结合“用户操作”与“设备状态”双维度进行采集。核心范围包括：用户操作数据(设备添加及删除、远程控制指令、参数调节记录、页面访问路径、停留时长、功能点击频率)、设备交互数据(设备在线及离线时间、响应指令耗时间、故障报警触发次数、用户对报警的处理方式)、业务场景数据(不同时段或区域的设备使用率、用户偏好设置(如常用温度、灯光亮度)、多设备联动场景的触发频率)。数据采集需遵循合规性原则，明确数据采集范围并告知用户，避免隐私泄露。

2. 核心分析维度与方法

(1) 行为路径分析。梳理用户从登录到完成操作的完整路径，识别高频路径与卡顿节点。例如，若多数用户添加设备时反复跳转页面，说明设备添加流程烦琐，需优化界面交互逻辑。常用分析方法为路径流程图、漏斗模型，用于量化各环节的转化率。

(2) 设备使用特征分析。统计设备使用率、使用时段、功能偏好等数据，提炼用户使用习惯。例如，分析智能家居设备数据发现，用户每日7:00自动开启窗帘、22:00关闭灯光，可优化“起床模式”及“睡眠模式”的默认参数，提升个性化体验。常用方法为描述性统计及时段分布直方图。

(3) 异常行为与故障关联分析。关联用户操作与设备故障数据，判断故障是用户误操作还是设备本身问题。例如，若多次设备离线均发生在用户切换Wi-Fi后，可推测是网络切换导致的连接异常，需优化设备网络重连机制。常用方法为关联规则挖掘、异常检测算法。

(4) 业务价值转化分析。针对商业场景，分析用户行为与业务指标的关联，如设备付费率、增值服务使用率。例如，工业物联网中，若某用户频繁使用数据报表导出功能，可推荐付费版高级分析服务。常用方法为相关性分析及用户分层画像。

3. 分析结果的应用落地

分析结果需转化为具体的优化动作，形成“采集—分析—优化—验证”的闭环：一是界面与功能优化，简化高频操作路径，隐藏低频功能，例如将用户常用的“设备控制”置于App首页；二是产品迭代，基于用户偏好新增功能，如多数用户需要多设备联动，可开发自定义场景功能；三是运维优化，提前预判设备故障风险，如某区域设备频繁离线，可

排查网络覆盖问题；四是个性化服务，为不同用户群体推送定制化内容，如为老年用户提供极简版控制界面。

2.4.3 可视化与报警通知机制

可视化与报警通知机制是物联网应用层“让数据说话、让异常可见”的核心手段，前者将抽象数据转化为直观界面，后者确保异常情况及时触达用户，二者结合可保障物联网系统的易用性与可靠性。

1. 数据可视化设计原则与实现

物联网数据可视化需遵循“精准、直观、高效”的原则，结合数据类型与使用场景选择合适的展示形式，避免冗余设计。核心设计要点如下。

(1) 数据分类展示。实时数据(如当前温湿度、设备状态)采用仪表盘、数字卡片展示，突出核心指标；趋势数据(如24小时温湿度变化)，采用折线图、面积图展示，清晰呈现波动规律；多维度数据(如多区域设备能耗)采用柱状图、热力图展示，便于对比分析；地理分布数据(如户外传感器位置)采用地图标注展示，直观呈现设备布局。

(2) 交互设计优化。支持图表缩放、时间筛选、数据钻取功能。例如，单击折线图某一节点可查看对应时间点的详细数据；支持多图表联动，例如选择某一设备时，所有图表同步展示该设备的数据。

(3) 视觉分层设计。采用颜色编码区分数据状态，如绿色表示正常、黄色表示预警、红色表示异常；通过字体大小、卡片样式突出核心数据，避免信息杂乱。

实现工具上，教学重点掌握ECharts的基础应用，通过调用API接口从云平台获取数据，动态渲染图表。例如，通过ECharts绘制温湿度趋势图，设置定时请求机制，实现数据实时更新。

2. 报警通知机制的核心构成

报警通知机制需确保“异常可感知、处理可追溯”，核心构成包括报警触发条件、通知方式、处理流程3大模块，形成闭环管理。

(1) 报警触发条件。分为阈值报警、状态报警、异常行为报警3类。阈值报警由用户预设参数范围，如温度高于30℃、湿度低于20%时触发；状态报警针对设备运行状态，如设备离线、通信故障及电量低时触发；异常行为报警针对用户误操作或恶意操作，如多次输入错误密码、频繁下发非法指令时触发。条件设置需支持自定义，以适配不同场景需求。

(2) 多渠道通知方式。采用“多渠道联动”确保通知及时触达，核心方式包括App推送通知(优先级最高，实时弹窗提醒)、短信或电话通知(针对紧急报警，如设备故障停机)、Web端弹窗或铃声提醒(针对管理员，适配监控场景)、邮件通知(针对非紧急报警，如数据异常日报)。同时支持通知分级，根据报警严重程度选择对应渠道，避免信息过载。

(3) 报警处理流程。设计标准化流程确保问题闭环，包括：报警接收(用户查看报警详情，如异常指标、发生时间、设备位置)、故障排查(系统提供排查建议，如设备离线可检查网络)、处理反馈(用户提交处理结果，如“已修复”或“需上门检修”)、报警归档(系统留存报警记录与处理日志，便于后续复盘)。部分场景可实现自动处理，如温度超标时自动触发空调开启，无须用户干预。

2.4.4　实例：开发一个简单的物联网监控 Web 界面

本实例以“温湿度实时监控”为核心业务场景，基于本科阶段重点掌握的Vue.js 及ECharts技术栈，搭建轻量化物联网监控Web界面，实现设备列表展示、实时温湿度可视化及阈值报警提示3大核心功能，帮助读者理解Web端开发的核心流程与落地逻辑。

1. 开发环境准备

(1) 基础环境：安装Node.js(建议16.x及以上版本)，自带npm包管理工具，用于搭建Vue项目和安装依赖。

(2) 开发工具：推荐VS Code，搭配Vetur(Vue代码高亮)及ECharts Helper(图表提示)插件，以提升开发效率。

(3) 依赖安装：需提前安装Vue脚手架及核心依赖包，终端执行以下命令：

```
//Bash
# 全局安装Vue脚手架(快速创建项目)
npm install -g @vue/cli
# 验证安装成功
vue --version
```

2. 技术选型与项目结构

(1) 前端框架：Vue.js 2.x(入门门槛低、文档完善)。

(2) 可视化工具：ECharts 5.x(支持多类型图表，适配物联网数据展示)。

(3) 数据层：本地模拟JSON数据(替代真实云平台接口，降低实操门槛)。

(4) 简化版项目结构(聚焦核心功能)：

```
//Plain Text
iot-monitor/              # 项目根目录
├── public/               # 静态资源
│   └── index.html        # 入口HTML文件
├── src/                  # 核心代码目录
│   ├── components/       # 自定义组件目录
│   │   ├── DeviceList.vue    # 设备列表组件(展示设备状态及实时数据)
│   │   └── TempHumChart.vue  # 温湿度趋势图表组件
│   ├── App.vue           # 根组件(整合所有子组件)
│   └── main.js           # 项目入口JS文件
└── package.json          # 依赖配置文件
```

3. 核心功能实现(部分关键伪代码)

(1) 创建Vue项目并安装依赖：

```
//Bash
# 创建项目(选择默认Vue 2模板)
vue create iot-monitor
# 进入项目目录
cd iot-monitor
# 安装ECharts(可视化)和axios(后续可对接接口)
npm install echarts axios --save
```

(2) 设备列表组件(DeviceList.vue)：展示设备ID、名称、在线状态、实时温湿度，温湿度超阈值时自动标红，直观提示异常。

```
//Plain Text
<template>
  <div class="device-list">
    <h3>物联网设备实时状态</h3>
    <table border="1" cellpadding="8" cellspacing="0">
      <thead>
        <tr>
          <th>设备ID</th>
          <th>设备名称</th>
          <th>在线状态</th>
          <th>温度(℃)</th>
          <th>湿度(%)</th>
        </tr>
      </thead>
      <tbody>
        <!-- 循环渲染设备列表 -->
        <tr v-for="device in deviceList" :key="device.id">
          <td>{{ device.id }}</td>
          <td>{{ device.name }}</td>
          <!-- 在线状态：绿色=在线，红色=离线 -->
          <td :style="{ color: device.online ? 'green' : 'red' }">
            {{ device.online ? '在线' : '离线' }}
          </td>
          <!-- 温度超30℃标红(阈值可自定义) -->
          <td :style="{ color: device.temp > 30 ? 'red' : 'black' }">
            {{ device.temp }}
          </td>
          <!-- 湿度低于20%标红(阈值可自定义) -->
          <td :style="{ color: device.hum < 20 ? 'red' : 'black' }">
            {{ device.hum }}
          </td>
        </tr>
      </tbody>
    </table>
  </div>
</template>

<script>
export default {
  name: "DeviceList",
  data() {
    return {
      // 模拟设备数据(实际项目中从云平台接口获取)
      deviceList: [
        { id: "dev001", name: "客厅传感器", online: true, temp: 28, hum: 45 },
        { id: "dev002", name: "车间传感器1", online: true, temp: 32, hum: 18 },
        { id: "dev003", name: "仓库传感器", online: false, temp: 25, hum: 30 },
```

```
            ],
          };
      },
  };
  </script>

  <style scoped>
  .device-list {
    margin: 20px auto;
    width: 90%;
  }
  table {
    width: 100%;
    border-collapse: collapse;
  }
  th {
     background-color: #f5f5f5;
  }
  </style>
```

(3) 温湿度趋势图表组件(TempHumChart.vue)：用双轴折线图展示“客厅传感器”近 12 小时温湿度变化，直观呈现数据波动规律。

```
//Plain Text
<template>
  <div class="chart-container">
    <h3>客厅传感器温湿度趋势(近12小时)</h3>
    <!-- ECharts容器，必须设置宽高 -->
    <div id="tempHumChart" style="width: 100%; height: 400px;"></div>
  </div>
</template>

<script>
// 引入ECharts核心库
import * as echarts from "echarts";

export default {
  name: "TempHumChart",
  // 页面挂载后初始化图表(Vue生命周期钩子)
  mounted() {
    this.initChart();
  },
  methods: {
    initChart() {
      // 获取DOM容器
      const chartDom = document.getElementById("tempHumChart");
      // 初始化ECharts实例
      const myChart = echarts.init(chartDom);

      // 模拟近12小时时间轴和数据
      const hourData = ["0点", "2点", "4点", "6点", "8点", "10点", "12点", "14点", "16点", "18点",
```

```
"20点", "22点"];
        const tempData = [22, 21, 20, 21, 24, 26, 28, 29, 28, 27, 26, 25]; // 温度数据
        const humData = [40, 42, 45, 44, 43, 41, 45, 46, 44, 42, 41, 40]; // 湿度数据

        // 图表配置项(核心)
        const option = {
          title: { text: "温湿度变化趋势" }, // 图表标题
          tooltip: { trigger: "axis" }, // 鼠标悬停显示数据详情
          legend: { data: ["温度(℃)", "湿度(%)"] }, // 图例(区分两条折线)
          xAxis: { type: "category", data: hourData }, // X轴：时间
          // 双Y轴：适配温度/湿度不同数值范围
          yAxis: [
            { type: "value", name: "温度", min: 0, max: 40 }, // 左侧Y轴(温度)
            { type: "value", name: "湿度", min: 0, max: 100, position: "right" } // 右侧Y轴(湿度)
          ],
          // 系列数据(折线图)
          series: [
            {
              name: "温度(℃)",
              type: "line",
              data: tempData,
              color: "#ff7f50" // 橙色标识温度
            },
            {
              name: "湿度(%)",
              type: "line",
              data: humData,
              yAxisIndex: 1, // 关联右侧Y轴
              color: "#1e90ff" // 蓝色标识湿度
            }
          ]
        };

        // 渲染图表
        myChart.setOption(option);
        // 适配窗口大小变化(保证图表响应式)
        window.addEventListener("resize", () => {
          myChart.resize();
        });
      },
    },
};
</script>

<style scoped>
.chart-container {
  margin: 20px auto;
  width: 90%;
}
</style>
```

(4) 根组件整合(App.vue)：将设备列表和图表组件整合，形成完整的监控界面。

```
//Plain Text
<template>
  <div id="app">
    <h2 style="text-align: center; margin: 20px 0;">物联网温湿度监控Web界面</h2>
    <DeviceList />
    <TempHumChart />
  </div>
</template>

<script>
// 引入自定义组件
import DeviceList from "./components/DeviceList.vue";
import TempHumChart from "./components/TempHumChart.vue";

export default {
  name: "App",
  components: {
    DeviceList,
    TempHumChart,
  },
};
</script>

<style>
#app {
  font-family: "Microsoft YaHei", sans-serif;
  color: #333;
  max-width: 1200px;
  margin: 0 auto;
}
</style>
```

4. 运行程序

使用以下命令启动本地开发服务器：

```
Bash
# 启动本地开发服务器
npm run serve
```

执行后，终端将输出访问地址(如http://localhost:8080)。在浏览器中访问该地址后，将看到以下内容。

(1) 设备列表：车间传感器1温度32℃ (标红)、湿度18%(标红)，仓库传感器离线(标红)，如图2-7所示。

物联网设备实时状态

设备ID	设备名称	在线状态	温度(℃)	湿度(%)
dev001	客厅传感器	在线	28	45
dev002	车间传感器1	在线	32	18
dev003	仓库传感器	离线	25	30

图 2-7　运行界面

(2) 温湿度趋势图：展示12小时温湿度折线变化，鼠标悬停可查看具体时间点的数值，如图2-8所示。

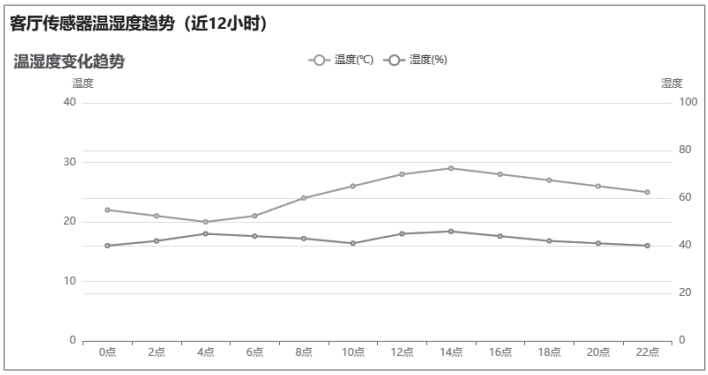

图 2-8　温湿度趋势图

5. 优化建议

(1) 接口对接：替换本地模拟数据，通过axios调用物联网云平台API(如GET/api/device/list获取设备列表)。

(2) 实时更新：引入WebSocket实现数据推送，模拟真实场景下“秒级更新”的监控需求。

(3) 报警增强：添加弹窗及提示音，温湿度超阈值时自动触发报警提醒。

(4) 权限控制：新增登录页面，区分“管理员与普通用户”，管理员可修改报警阈值。

2.5 本章小结

本章在物联网三层架构基础上，进一步将应用层拆解为平台与云服务层、应用与用户交互层，构建了感知层、网络传输层、平台与云服务层、应用与用户交互层的四层架构体系，并围绕各层级展开了理论与实践思路的讲解，帮助读者掌握物联网系统的组成逻辑与工程应用能力。

感知层作为物联网的“感知器官”，是数据采集的源头，核心组件包括温湿度、光敏、加速度等各类传感器，RFID识别设备及嵌入式系统(如Arduino、ESP32)，本章通过Arduino采集温湿度的实践思路，讲解了感知层设备选型、接线与基础开发，强调了数据采集精准、高效、低功耗的核心要求。

网络传输层是“神经网络”，承担数据双向传输功能，涵盖有线与无线两类通信方式。重点介绍了Wi-Fi、蓝牙、ZigBee、LoRa、NB-IoT等主流无线通信技术的特性与选型原则，并通过ESP32连接Wi-Fi上传数据至Thingspeak平台的实践，详细讲解了感知层与网络层的数据交互逻辑。

平台与云服务层是“大脑中枢”，实现设备管理、数据存储与处理、消息队列分发等功能。本章介绍了AWS IoT Core、阿里云IoT、华为云IoT等主流云平台，解析了设备全生命周期管理与MQTT消息队列机制，并以ESP32接入阿里云IoT平台为例，讲解了设备认证、数据上传与远程控制的完整流程。

应用与用户交互层是“落地载体”，聚焦数据可视化、用户交互与价值转化。本章讲

解了移动App与Web端的开发选型、用户行为数据分析方法，以及可视化与报警通知机制，并通过Vue+ECharts开发温湿度监控Web界面的实践思路，展现了应用层从数据呈现到异常预警的核心能力。

四层架构并非孤立存在，而是形成“感知—传输—处理—交互”的闭环逻辑：感知层采集数据，经网络层传输至平台层处理分析，再由应用层呈现给用户并接收指令反向控制感知层设备。本章通过理论与实例相结合，不仅梳理了各层级的核心设备、技术原理与功能定位，更培养了物联网系统的工程思维，形成“需求→选型→开发→落地→优化”的实操闭环，为后续核心技术学习与项目实战奠定了基础。

2.6 思考与练习

一、选择题

1. 下列不属于物联网移动App核心功能的是(　　)。

A. 设备列表管理　　B. 多用户权限分配　C. 个性化场景设置　D. 实时数据展示

2. 适合展示“24 小时温湿度波动规律”的图表类型是(　　)。

A. 仪表盘　　B. 折线图　　C. 热力图　　D. 饼图

3. 用户行为分析中，用于识别“设备添加流程卡顿节点”的方法是(　　)。

A. 漏斗模型　　B. 描述性统计　　C. 关联规则挖掘　　D. 用户分层画像

二、简答题

1. 简述物联网移动App与Web端的功能互补性，并举例说明二者在智能家居场景中的协同逻辑。

2. 物联网报警通知机制包含哪三大核心模块？简述“阈值报警”的完整处理流程。

3. 为什么说用户行为数据分析是“数据驱动优化”的核心？列举两个物联网场景下的分析落地案例。

三、实践题

1. 基于本章实例，为Web界面新增“报警阈值设置”功能：管理员可修改温度(默认30℃)、湿度(默认20%)的报警阈值，修改后设备列表的标红逻辑同步更新。

2. 基于Flutter或React Native，设计极简版物联网App：展示1个智能灯设备，单击开关按钮切换设备状态(开/关)，并实时更新界面显示。

3. 设计一份“智能家居App用户行为数据采集清单”，涵盖“用户操作、设备交互、业务场景”3类数据，每类至少列出5项具体指标。