物联网应用开发与生成：从低代码到智能原生

束方磊，刘佳伟，李博睿，王帅
（东南大学计算机科学与工程学院，江苏 南京 211189）

摘 要：物联网（IoT, Internet of things）技术通过设备、传感器和网络的互联，实现了物与物、物与人之间的智能交互和数据共享，已广泛应用于智能家居、智慧交通和工业自动化等领域。传统的物联网开发流程依赖于人工的软硬协同开发，具有较高的技术门槛。低代码开发技术通过图形化编程界面和高度抽象的编程接口，显著地降低了开发门槛，但仍存在定制化能力不足的问题。随着以大语言模型为代表的人工智能（AI, artificial intelligence）及其相关技术的成熟，一种利用人工智能模型对物联网应用开发流程进行高层语义表征的“智能原生”物联网计算任务生成范式逐渐兴起，成为物联网应用开发的新机遇。为此，在梳理物联网应用开发技术的发展的基础上，提出了智能原生物联网计算任务生成框架，将开发过程划分为物联网应用计算意图解析与计算规划生成两个阶段。在此基础上，系统性地分析了智能原生物联网应用生成的关键技术和挑战，探讨了最新研究进展，最后对未来发展方向进行了展望。

关键词：物联网；智能原生计算；低代码开发；大语言模型
中图分类号：TP393.5；TP311.5
文献标志码：A
doi：10.11959/j.issn.2096−3750.2026.00510

IoT application development and generation: from low-code to AI-native

Shu Fanglei, Liu Jiawei, Li Borui, Wang Shuai
School of Computer Science and Engineering, Southeast University, Nanjing 211189, China

Abstract: Internet of things (IoT) enables intelligent interaction and data sharing, facilitating connectivity among objects and humans. It has been widely applied in smart homes, intelligent transportation, industrial automation, etc. Traditional IoT development processes rely on manual hardware-software collaborative development, which involves a high technical barrier. Low-code development technology significantly lowers this barrier by providing graphical interfaces and highly abstracted programming application programming interfaces. Nevertheless, it still faces limitations in customization capabilities. With the maturity of artificial intelligence (AI) and related technologies represented by large language models, an emerging AI-native paradigm for IoT computing task generation characterized by leveraging AI models to provide high-level semantic representations of IoT application development workflows is gaining traction and presenting novel opportunities for IoT application development. Therefore, on the basis of reviewing the development of software development technology for the IoT, an AI-native framework for generating computing tasks of IoT was proposed in this paper, which was divided into two stages: intent understanding and task planning. Based on this, the key technologies and challenges were systematically analyzed, a comprehensive review of the latest research was provided, and the future development directions of AI-native computing task generation were finally looked forward to.

Key words: IoT, AI-native computing, low-code development, large language model

收稿日期：2025−02−26；**修回日期：**2025−06−17
通信作者：王帅，shuaiwang@seu.edu.cn
基金项目：国家自然科学基金项目（No. 62302096，No. 62272098，No. U24B20152）
Foundation Items: The National Natural Science Foundation of China (No. 62302096, No. 62272098, No. U24B20152)

0　引言

物联网（IoT, Internet of things）是在互联网、传感器技术、通信技术等多领域快速发展的背景下兴起的新兴技术，它通过将设备、传感器和网络连接起来，实现物与物、物与人之间的智能化交互和数据共享。随着 5G、大数据和云计算等技术的成熟，物联网的应用场景不断拓展，从智能家居、智慧交通到工业自动化等领域，正深刻地改变着人们的生活和生产方式，推动全球进入智能化时代[1-2]。

传统的物联网应用开发流程通常包括以下步骤：1) 开发人员对功能需求进行分析，拆解功能模块，并与物理设备功能进行匹配；2) 针对不同的功能模块，在不同硬件设备上使用不同的编程语言进行编程开发；3) 将各个应用模块部署至物联网设备，或部署至云端和边缘设备以完成协同计算[3-4]。上述开发流程高度依赖人工，并且需要软硬协同、多端协同开发，对开发人员的技术能力提出了较高的要求。

针对此问题，研究人员提出了物联网低代码开发技术[5-8]，通过高度抽象的编程接口定义与图形化编程界面，简化物联网应用开发过程。用户可以通过拖拽组件、配置流程等方式表达应用逻辑，低代码开发系统自动生成物联网应用。物联网低代码开发能够减少手动编码工作，从而提高开发效率，降低开发门槛[9]。但是，物联网低代码开发仍然需要开发人员具有一定专业知识，具备使用低代码开发工具清晰表达出编程逻辑的能力。并且，开发人员通过低代码方式开发出来的应用受低代码平台的编程抽象制约，难以实现定制化、个性化的应用。

随着人工智能技术的快速发展，越来越多的人类工作被人工智能模型所取代，例如，自动驾驶、语音识别、图像识别等[10-13]。在此基础上，是否可以通过人工智能模型来表征物联网应用的开发流程，从而替代人工实现全流程的智能化物联网应用开发，是未来物联网应用开发的一个重要趋势和发展方向。本文将这种使用人工智能理论重塑物联网应用生成这一基本物理信息系统功能的方式称为“智能原生”计算任务生成。

近年来，大语言模型在生成与推理能力上的不断成熟为“智能原生”下的计算任务生成提供了新的机遇。大语言模型对物联网开发的变革体现在两个维度：一是依托强大的规划能力，智能体可自主替代人力完成复杂的任务逻辑构建[14-16]；二是基于预训练积累的广泛先验知识[17-19]，这种知识积累使大语言模型具备较强的零样本（zero-shot）场景适应性。开发人员仅须对大语言模型进行简单地微调或补充特定领域的语料，就能使模型适应新环境，满足不同场景的需求。相比于传统依赖人工的物联网开发以及低代码开发等方式，智能原生计算应用生成能够自动理解用户需求、理解场景并生成对应计算任务。这种转变显著地提升了物联网应用开发效率，实现了物联网应用开发的全流程智能化，同时确保了所生成的应用能够有效地应对多样化的场景需求。

目前，国内外关于大语言模型与物联网技术结合的综述文章主要集中于以下几个方面：文献[20-22]系统地分析了基于大语言模型的智能体框架、发展与应用；文献[23-24]详细地阐述了大语言模型在智能体规划方面的相关技术；文献[25]探讨了大语言模型在具身智能领域的应用；文献[26]分析了大语言模型在多模态任务中的实际应用与潜力；文献[27-28]系统地阐述了大语言模型结合检索增强生成技术的相关研究进展。尽管上述文献对大语言模型在特定应用场景中的技术细节进行了详尽探讨，但针对其在智能原生计算任务生成的研究仍缺乏系统性总结和分析。

本文旨在全面地阐述面向物联网的智能原生计算任务生成的理论基础与技术进展，并系统性地概括了计算任务生成过程中的关键技术。本文将智能原生计算任务生成划分为物联网计算意图解析和物联网计算规划生成两个主要阶段，深入地分析了各个阶段相关关键技术及面临的挑战，并对智能原生计算任务生成的未来发展方向进行了展望。本文的主要贡献为以下 3 个方面。

1) 将传统物联网编程开发与智能原生下的计算任务生成进行了对比，提出了智能原生计算任务生成框架。该框架将整个生成过程划分为计算意图解析和计算规划生成两个主要阶段。

2) 基于提出的框架，阐明了近年来大语言模型在物联网场景中的技术应用，深入探讨了其在智能原生计算任务生成中的最新进展。

3) 综合分析了智能原生计算任务生成的研究趋势，总结了其进一步发展与推广所面临的技术挑战。

1　物联网应用开发的发展历程

本节简要地回顾了传统物联网编程开发流程和物联网低代码开发，并探讨了智能原生下的计算任务生成，旨在展示从传统开发模式向智能化转变的过程。

1.1　传统物联网编程开发

物联网应用系统架构从上至下通常分为3层：云计算层、边缘计算层以及物联网节点层。云计算层通常由服务器组成，能够提供远高于边缘设备和物联网设备的计算能力，因此，其主要负责处理海量数据并训练人工智能模型，同时向下提供必要的控制接口；边缘计算层包含无线网关、无线基站等，作为中间层一方面能够帮助物联网设备接入公共骨干网络，另一方面能够执行简单的本地计算，对上传至云端的数据进行预处理，同时传递来自云计算层的指令[29]；在物联网架构中，节点层涵盖了所有部署于一线环境的感智设备，其核心功能在于多源数据的采集、即时运算以及对边缘控制命令的反馈。不仅如此，节点还可通过动作执行单元完成与外界或人类的深度对接，实现闭环交互[30-32]。

物联网应用开发的3种不同方式如图1所示，传统物联网编程开发如图1(a)所示。首先，开发人员须分析用户需求并将功能模块按层级拆分；接着，手动编写各层级代码；最后，进行联合调试。此过程虽然将各个层级的任务进行了拆解，但各端之间的代码仍须考虑相互之间的交互，以实现交换数据或指令下达。传统物联网应用的编程开发不仅要求开发人员掌握多种编程语言，还须具备相关领域的专业知识，导致学习成本较高、开发效率低下。此外，由于开发过程中须编写大量代码，进一步增加了开发周期和错误风险。

1.2　物联网低代码开发

低代码开发是近年来兴起的新型开发模式，借助可视化界面和少量编程代码，用户可以快速构建并部署应用程序[33-34]。低代码平台通过自己开发的接口，将云边融合的3层架构中的各层进行融合，物联网低代码开发如图1(b)所示。在这些平台中，开发人员可以通过拖放组件的方式创建应用程序，减少了对复杂编程技能的需求[35-36]。EdgeProg [37]在低代码开发的基础上提出了一种以边缘为中心的物联网应用一体化编程模型。该模型使用户可以在边缘服务器上编写处理逻辑，边缘服务器会自动生成应用代码，并自动将代码分割成设备代码和服务器代码，以实现最佳延迟。此外，物联网场景中除了软件逻辑编程外，还须考虑底层物联网硬件的选型和连接问题。TinyLink[5]采用自顶向下的方法来设计物联网应用的硬件和软件。TinyLink提供统一的应用程序接口，供应用与底层硬件组件交互。开发者通过类似C语言的代码编写应用程序，指定应用的关键逻辑，而无须处理具体硬件组件的细节。TinyLink以应用代码为输入，自动生成硬件配置以及在目标硬件平台上可执行的二进制文件。AutoLink[38]提出了面向多性能指标的物联网硬件平台的按需生成方案，用户输入功能需求以及约束条件，系统即可自动完成硬件选型，极大地简化了开发者的工作。借助于对云端开发模块的进一步集成，TinyLink 2.0[6]完成了从感知节点到云端算力的全链路开发覆盖。其配套的编程语言旨在简化逻辑表达，助力开发者高效地部署多样化的物联网应用。虽然这种低代码方案在成本控制和简化流程上表现优异，但操作自由度仍存在局限性，专业知识背景仍是高质量开发的重要前提。

1.3　智能原生下计算任务生成

随着大语言模型及其衍生的智能体（agent）的快速发展，物联网应用开发迎来了新的机遇。大语言模型智能体凭借强大的语言理解、任务规划、泛化生成和工具使用能力，有望在智能原生愿景中自动化整个应用开发流程。智能原生计算生成如图1(c)所示，大语言模型智能体将接管从需求分析到联合调试的整个应用开发流程，自动完成所有开发任务，生成所需的计算任务。其中，智能体能够通过检索外部知识库或对自身进行微调，确保生成的代码能够适应特定的场景和任务需求。这种方式大幅地减少了开发人员的参与度，开发人员仅须明确表达需求，智能体就能自动生成适配场景的程序代码，实现全流程、全场景的智能化。

1) 物联网应用计算意图解析：意图解析指的是智能体在接收到用户需求时，使用大语言模型来解析用户的编程需求。这不仅包括对用户输入的文本需求的解析，还涉及对物理世界多模态信息的解析[39-41]。在智能原生中，除了解析用户输入的文本需求，还需要大语言模型去解析物理世界的多模态信息。现有的研究主要集中在两个方向：一是将大

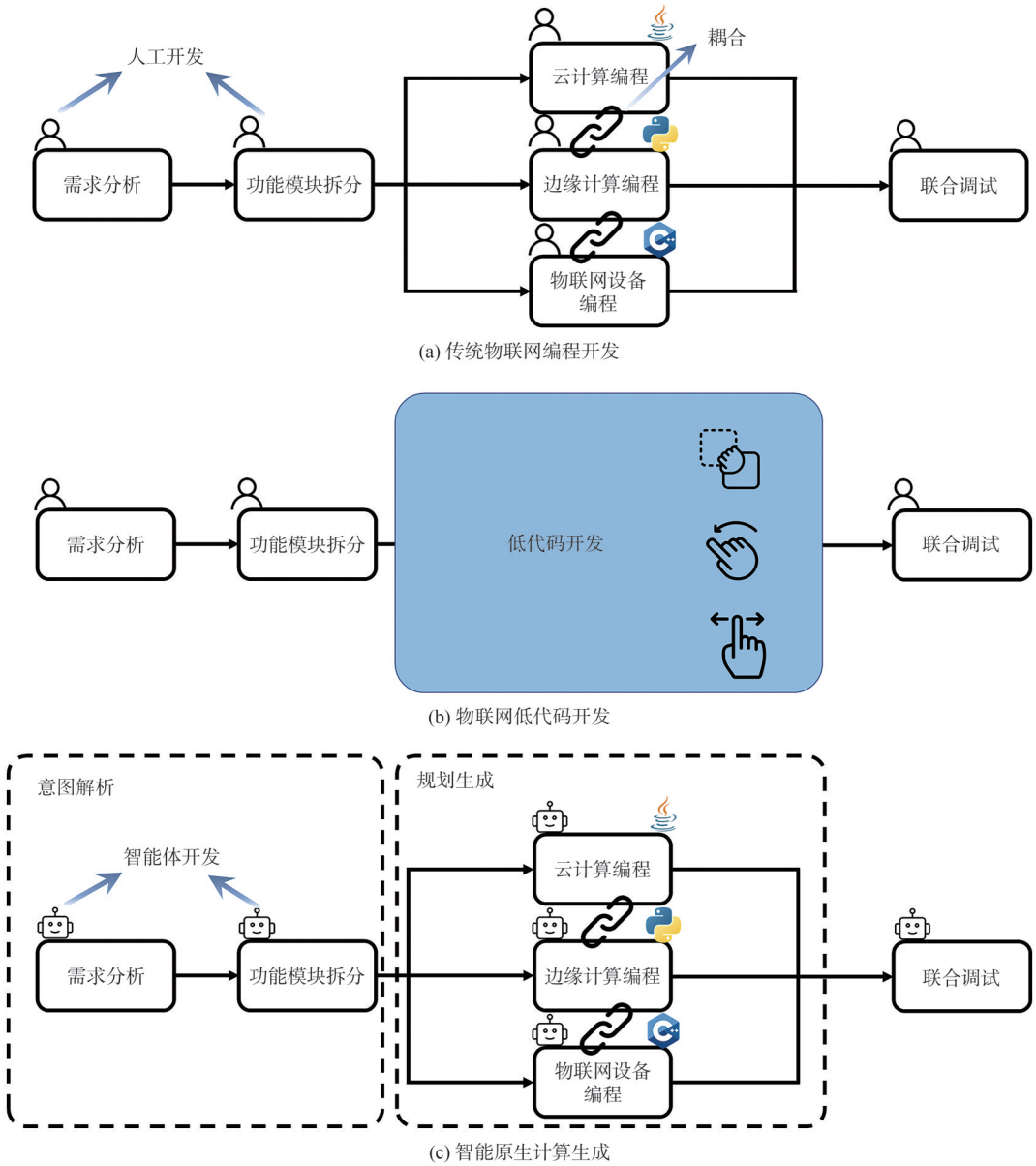

图 1　物联网应用开发的 3 种不同方式

型语言模型设计为能够直接识别多模态信息的架构；二是利用外部工具将多模态信息转化为文本信息，随后将其传输至大型语言模型进行处理。

2) 物联网计算任务规划生成：规划生成是指在解析完用户需求后，利用大语言模型生成所需要的代码。在这一阶段，智能体通过引入规划增强技术，进一步提高任务生成的准确性和适应性。物联网计算规划生成涵盖智能原生物联网计算任务生成和计算任务生成范式两个部分。在此阶段，大语言模型智能体根据解析后的用户意图，生成适用于特定场景任务的通用编程语言或特定领域计算代码。同时，通过无反馈和有反馈规划生成范式，进一步提高任务规划的准确性和适应性。这一过程不仅优化了任务调度，还提升了系统的整体性能和响应能力。

本文重点讨论面向物联网的智能原生计算任务生成相关研究工作，并从多个维度对现有代表性方法进行归纳与比较，面向物联网的智能原生计算任务生成相关工作见表 1。

2　物联网应用计算意图解析

意图解析是指大语言模型智能体利用预训练模型解析用户输入数据，识别用户任务意图。在物联网场景中，智能体接收的用户意图可能来源于多种模态，包括文本、语音，以及毫米波雷达、惯性传

表 1　面向物联网的智能原生计算任务生成相关工作

阶段	内容	方法及相关工作	含义
物联网应用计算意图解析	基于基础模型的计算意图解析	单一模态编码器、一体化基础模型，文献[42-46]	智能体所使用的大模型本身能够解析物联网多模态输入
	基于模型编排的计算意图解析	代码调用工具、指令调用工具，文献[47-49]	智能体通过调用其他工具模型来解析物联网多模态输入
物联网计算任务规划生成	智能原生物联网计算任务生成计算任务规划生成范式	面向通用编程语言的生成，文献[50-55]	智能体所生成的计算任务，其编程语言是通用编程语言
		面向特定领域语言的生成，文献[56-61]	智能体所生成的计算任务，其编程语言是面向特定领域的
		无反馈计算任务生成，文献[62-66]	智能体在生成解决方案过程中，不会基于之前的执行结果修正之后的规划
		基于环境反馈的计算任务生成，文献[67-71]	智能体在生成解决方案过程中，会基于之前的执行结果修正之后的规划

感器等。根据处理方式的不同，大语言模型智能体可以采用基于模型的方法和基于工具的方法进行意图解析。

2.1　基于基础模型的计算意图解析

大语言模型智能体识别多模态数据的第一种方式是通过模态编码器将不同模态的输入转换为模型可理解的特征表示，借助多模态基础模型的能力完成意图解析。在物联网场景中，图像包含大量的信息，是场景信息的重要来源之一。Vision Transformer（ViT）[42]将 Transformer 架构应用于图像处理，通过将图像分割为小块，并进行线性投影和多层 Transformer 编码，以提取图像特征。BEATS[43]使用声学分词器将连续音频信号转换为离散标签，支持掩码与离散标签预测预训练，为音频和多模态预训练奠定了基础。X-LLM[44]通过 X2L 接口（其中，“X”表示图像、语音和视频等多模态信息，“L”表示语言文本）将多模态内容（如图片、语音和视频）转换为文本后，输入到大语言模型进行处理。但是，物联网场景中，物联网应用通常无法同时处理高度多样化移动模态任务，而上述模型仅能够处理解析某种单一模态的信息，或者仅能应对极为有限的多模态任务。M4[45]是针对移动端多样化感知需求而设计的全能型基座模型，其系统架构由跨模态嵌入层、语义处理主干以及任务生成模块 3 部分组成。在该架构中，嵌入层负责将图像、语音及文本等异构输入映射为统一的标准化特征向量；核心主干则依托预训练大语言模型的深度推理能力完成数据解析；最后由生成模块根据应用场景将处理结果转化为特定格式。这种设计显著地增强了物联网终端处理复杂信息的能力，使其能在保持高效运行的同时，充分利用大语言模型的语义理解优势产出匹配结果。mPnP-LLM[46]采用了类似的思路，与 M4 不同的是，mPnP-LLM 将所有相关模态的编码器投射层接入到大语言模型的最后几层。这种设计使大语言模型不仅能够实现模态自适应，还能够显著地降低计算量。

2.2　基于模型编排的计算意图解析

物联网场景中，大语言模型智能体识别多模态数据的第二种方式是将多模态小模型作为工具。根据物联网场景的具体任务需求，智能体编排适合的多模态模型来处理特定任务，从而实现对多模态信息的意图解析。ViperGPT[47]使用大语言模型作为主控引擎，通过生成 Python 代码调用视觉模型和语言模型解决复杂的视觉理解任务。InternGPT[48]在处理视觉任务时，通过调度多模态模型识别指向性动作（如手势和光标），从而提供灵活且精确的控制，尤其适用于需要细粒度控制、编辑和生成视觉内容的场景。HuggingGPT[49]将大语言模型与机器学习社区（如，Hugging Face）中的人工智能模型相连接，用于理解用户意图。它通过 ChatGPT 进行任务规划，从 Hugging Face 中选择适合的多模态模型执行子任务，并基于执行结果生成响应。

3　物联网计算任务规划生成

本节深入地探讨了智能原生计算任务生成的第二阶段——物联网计算规划生成，特别是智能体如何在解析物联网场景的输入信息后生成相应的解决方案。该阶段进一步细分为两部分：智能原生计算任务生成和计算任务的生成范式。智能原生计算任务生成流程示意图如图 2 所示，用户将物联网应用的任务需求和场景信息输入大语言模型智能体，智能体进行规划生成。智能体的输出主要分为两类：

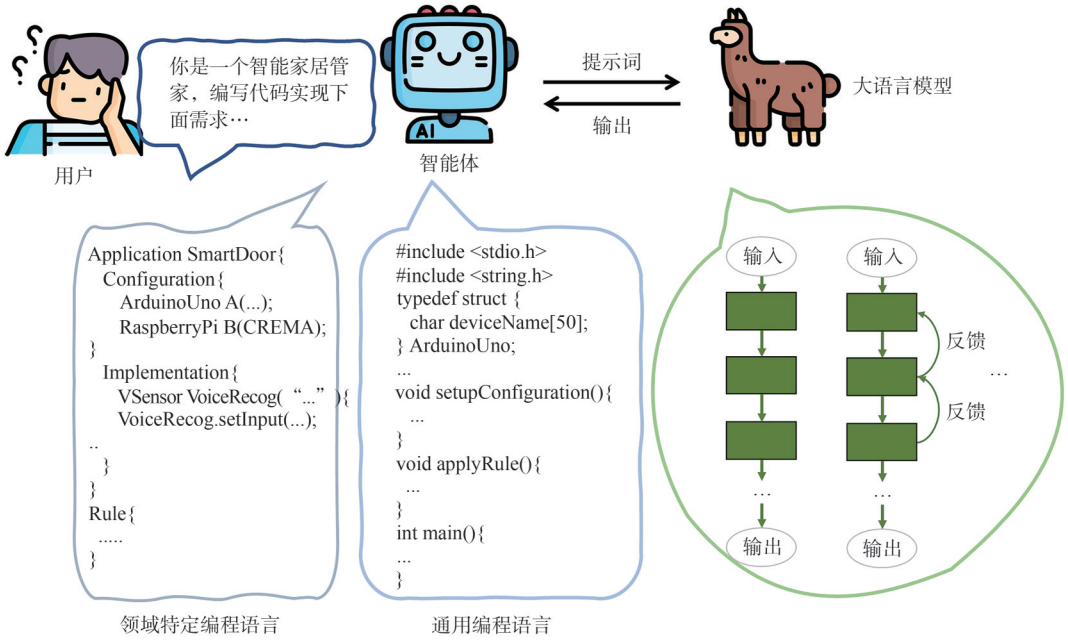

图2　智能原生计算任务生成流程示意图

面向通用编程语言和面向领域特定编程语言，生成范式则涵盖基于环境反馈的计算任务生成和无反馈计算任务生成两种。

3.1　智能原生物联网计算任务生成

在智能原生计算任务生成的流程中，大语言模型智能体识别完任务场景信息以及任务需求之后，随即生成对应任务的代码，这一阶段是智能原生计算任务生成的主要内容。由于计算任务执行场景的多样性，所需的执行代码也因物联网应用场景的不同而变化。因此，研究人员致力于引导大语言模型生成能够适应不同需求的代码语言。结合现有研究，本文将相关工作细分为两类：面向领域特定编程语言（DSL, domain-specific programming language）和通用编程语言（GPL, general-purpose programming language）的计算任务生成。

3.1.1　面向通用编程语言的计算任务生成

通用编程语言设计用于处理多种任务并应用于多个领域。ChatDev[50]利用大型语言模型来促进软件设计、编码和测试阶段的合作，最终生成符合用户需求的 Python 代码解决方案。MetaGPT[51]模拟了一个软件公司内部的角色分配（如产品经理、架构师、项目经理、工程师和质量保证工程师），实现从需求分析到 Python 代码生成、测试和部署的整个软件开发流程。

现有的大语言模型主要设计用于生成简单和通用的算法，但并非专门针对 IoT 应用。为填补这一空白，IoTCoder[52]使用了 3 个本地部署的小型语言模型（SLM, small language model），设计了一个专门为物联网应用开发合成程序的编程助手。IoTCoder 使用的 3 个小型语言模型分别是任务分解 SLM、需求转换 SLM 和模块化代码生成 SLM。任务分解 SLM 主要是将复杂的物联网应用分解为多个任务，并提供详细的描述；需求转换 SLM 将自然语言描述的分解任务转化为结构良好的规格说明；模块化代码生成 SLM 根据任务规格生成模块化代码。

除了使用预训练的特定的本地 SLM 之外，更通用的方式是使用大语言模型来生成物联网应用。面向通用编程语言的计算任务生成流程示意图如图 3 所示，大语言模型智能体首先需要解析用户输入及场景信息，之后将任务进行解耦，并从代码库中搜索相关代码，接着通过多个特定智能体协作生成对应的代码。大语言模型在自然语言生成代码方面展现出了一定的潜力，但在物联网代码生成领域却表现欠佳。究其原因，主要是大语言模型对嵌入式物联网代码的上下文理解不够深入，从而导致其在生成代码时，常常忽视操作系统特定的应用程序接口（API, application program interface）甚至出现与之冲突的情况。IoTPilot[53]是一个基于大语言模型的多代理物联网编程框架，设计了一种基于聚类

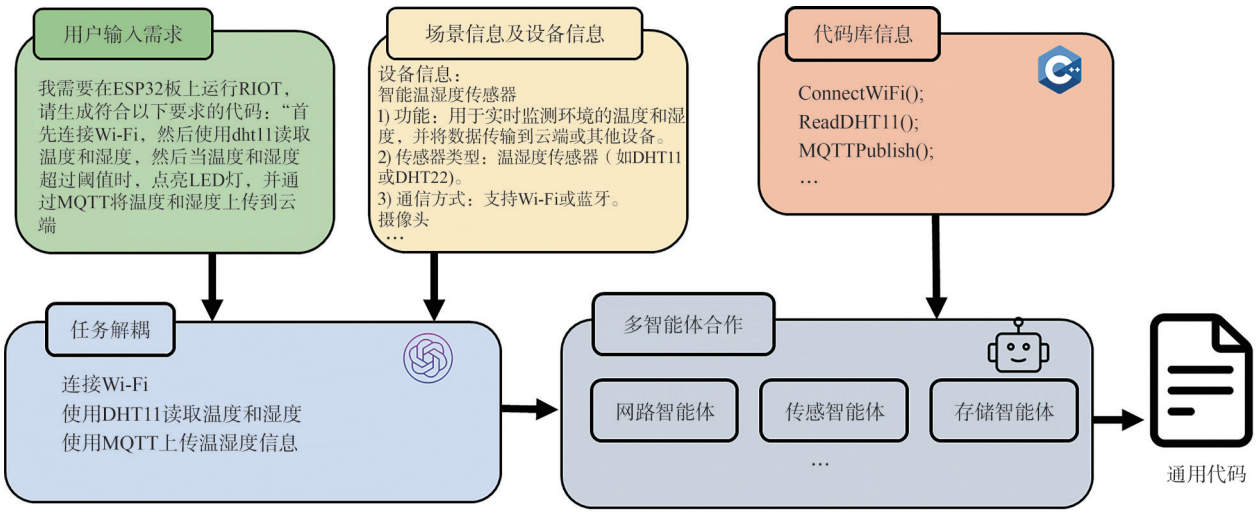

图3　面向通用编程语言的计算任务生成流程示意图

的渐进式检索增强生成策略（一种寻找与任务需求相关代码的策略）和自动校准的自我调试机制，以提高生成物联网应用的质量。具体来讲，IoTPilot为物联网代码生成任务设计了多个智能体，每个智能体负责处理特定的功能，如传感器、网络和存储。IoTPilot接收到用户需求描述之后，首先，会通过需求解析智能体将任务需求解耦成多个子任务（网络、传感器、存储等），然后，将子任务分配给对应功能的智能体。对应的智能体通过聚类的渐进式检索增强生成（RAG, retrieval-augmented generation）策略搜索操作系统的代码库来找到相关联的代码接口，随后进行几轮调试，以解决各个小功能模块的问题，如，网络或传感器组件。然后，它将这些完成的小功能模块整合成一个完整的最终代码。

在物联网领域，除了生成对应的代码，还需要解决硬件依赖问题。EmbedGenius[54] 利用大型语言模型的推理能力和嵌入式系统的专业知识，自动化硬件环路开发过程。EmbedGenius主要分为准备和执行两个阶段。在准备阶段，EmbedGenius会收集并标准化与硬件信息相关的基本元数据，并通过名称匹配、版本数量以及架构兼容性选择对应的硬件和库。最后基于库的内容生成对应的知识增强大语言模型。在执行阶段，EmbedGenius查询增强记忆的大语言模型以生成任务提示并自动化系统编程。此外，在传统的物联网开发过程中，用户须自定义一些规则。部分开发平台为简化用户开发流程，常采用预定义策略。然而，这些策略通常较为简单，易引发大量冲突，而监测并解决这些冲突反而给用户带来了额外的负担。因此，AutoIoT[55]基于大语言模型帮助用户生成无冲突的自动化规则，并帮助开发者生成冲突检测的代码，从而提升了用户体验。AutoIoT还设计了一个代码适配器，将逻辑推理与代码生成的语法细节分离，使大语言模型能够生成其训练数据之外的编程语言代码。

3.1.2　面向特定领域语言的计算任务生成

领域特定编程语言（SQL、IFTTT[56]等）是专为特定应用领域设计的编程语言，语法和功能经过优化以更好地满足该领域的特殊需求。与通用编程语言（Python、Java等）相比，DSL更简洁、易用，显著地提升了特定场景领域任务的开发效率。目前已有一些工作借助大语言模型生成特定领域的代码。例如，Chat-DB[57]和DB-GPT[58]指导大语言模型生成SQL语句，实现对私有数据库的操作。Chat-Excel[59]利用大语言模型生成Excel命令，实现了表格操作的自动化。

在物联网家庭自动化场景中，触发–动作程序（TAP, trigger-action program）是一种简单但强大的格式，用于实现智能物联网应用。传统的工作方法依赖于定制化的交互命令或标注良好的数据集，导致适用场景有限。ChatIoT[56]则引入了大语言模型来驱动TAP的零代码生成。ChatIoT代码生成流程示意图如图4所示，该系统的核心逻辑始于对用户意图及环境多模态数据的深度解析。为了优化输入，系统通过相似度计算筛选关联设备，从而实现提示词的精简。随后，由4个职能各异的智能体

（预处理、服务构建、代码生成与性能评估）协同完成 TAP 任务：预处理明确所需服务，服务构建负责补齐本地缺失的模型，代码生成输出具体逻辑代码，性能评估对代码质量进行审核。与 ChatIoT 类似，Sasha[60]智能家庭助手能够通过大语言模型解析用户输入的模糊指令，进而生成对应控制指令，响应用户的智能家居操作需求。与之前两个工作不同，SAGE[61]通过动态构建的大语言模型提示树来控制执行动作序列，并决定后续采取什么行动、某个动作是否成功以及何时终止过程。SAGE 的动作集增强了大语言模型的能力，并支持了智能家居助手的一些关键要求。这些要求包括：灵活且可扩展的用户偏好管理、通过 API 读取访问任何智能设备的完整功能，无需设备特定代码、持续的设备状态监控、仅通过房间照片进行自然设备引用等。

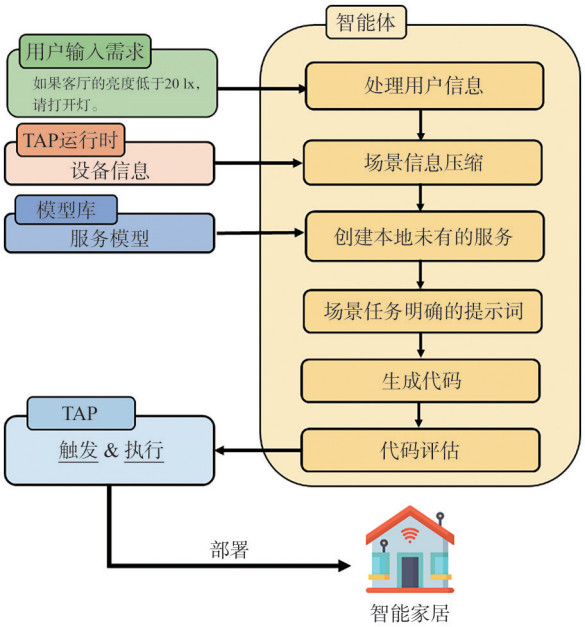

图 4　ChatIoT 代码生成流程示意图

3.2　计算任务规划生成范式

在智能原生计算任务生成中，大语言模型智能体在解析完输入的物联网场景信息后，随即规划并生成对应的解决方案。尽管大语言模型具备强大的语义理解能力，但随着物联网应用范围的扩大和任务复杂性的增加，大语言模型在处理复杂问题时的表现仍有待提升。为此，学术界提出了一系列方法来强化大语言模型智能体的规划能力。本节根据智能体是否具备对自身行为的反思能力，将相关工作划分为无反馈计算任务生成和有反馈计算任务生成两类。

3.2.1　无反馈计算任务生成

无反馈生成是指智能体不会对自身行为进行反思，即当前行为结果不会对后续规划产生任何影响。在智能原生计算任务生成过程中，采用无反馈生成的大语言模型智能体在接收任务需求和场景信息之后，就会进行逐步推理。在推理过程中，智能体并不会接受外部信息的变化，执行效率较高。因此，这种生成方式适合简单物联网场景应用的生成。

早期研究表明，通过提供少量示例[62]可以显著地提升大语言模型在 IoT 场景中的推理能力。思维链（CoT, chain of thoughts）[63]概念的引入—即通过将问题拆解为多个子任务并逐步解决—进一步提高了大语言模型在物联网任务中的效率，尤其是在处理复杂传感器数据和系统交互时。零样本思维链（zero-shot-CoT）[64]展示了仅须在问题结尾添加“让我们一步一步思考”即可引导大语言模型生成解决问题的思维链，而无须提供复杂示例。

然而，上述无反馈生成方法通常只能生成单一的思维路径，可能导致次优解或错误结果。为此，研究人员提出了多路规划方法，生成多个结果并进行综合评估，以得出最优解决方案。例如，CoT-SC[65]改进了传统的 CoT 方法，通过探索更多思维路径来更准确地捕捉相同问题下的多样性。在物联网应用生成过程中，CoT-SC 能够生成多个思维链，并通过多数投票法确定最终答案，进而提高系统决策的准确性和鲁棒性。思维树（ToT, tree of thoughts）[66]可视为 CoT 的延伸范式，核心是将大语言模型的推演机制转化为在树形解空间内的检索行为。该方法通过将复杂逻辑逐级剥离为微观子任务，并对潜在解路径进行动态评估与存储，在处理物联网领域的异构数据集成及多维任务编排时展现出显著优势。然而，受限于所依赖的深搜与广搜算法，ToT 通常伴随着高昂的算力开销与计算时延。

3.2.2　基于环境反馈的计算任务生成

无反馈计算任务规划生成的研究虽然在一定程度上增强了大语言模型的推理能力，但因缺乏与外界的交互，容易引发幻觉现象并导致错误的累积。尤其是在复杂多变的物联网场景中，场景信息时刻变化，大语言模型智能体的行为也会对场景信息进行改变。因此，大语言模型智能体应当时刻与外界场景或环境信息进行交互反馈，生成适配当前场景的物联网应用。基于环境反馈的计算任务生成方式

允许大语言模型智能体能够基于其之前的行动结果以及外部信息进行反思，以此指导后续的行为决策。ReAct[67]框架作为一种结合推理与行动的通用范式，适用于多种语言推理和决策任务。ReAct提示大语言模型生成并执行计划，根据执行结果动态地调整后续行动计划，同时通过与外部环境的交互引入新的知识。Self-refine[68] 侧重于挖掘大语言模型的自我更迭潜力，通过构建闭环反馈机制引导模型执行多轮自我修正，以此强化逻辑推理深度。Least-to-Most Prompt[69]采取了一种循序渐进的策略，将宏观复杂任务拆解为具有逻辑关联的微观子任务，并利用已解决环节的结论为后续推演提供支撑。

上述反馈生成的研究主要依赖于大语言模型自身的反馈机制来进行改进。然而，在物联网场景中，由于场景的特定性，模型生成的解决方案通常缺乏复合场景的专业检查机制，容易在错误的路径上不断迭代。因此，一些研究工作探索了借助外部工具提供反馈的方法。例如，作为ReAct范式的进阶，Reflexion[70]通过集成外部审核模块来捕捉逻辑偏差，并利用反馈信号引导大语言模型执行纠正。CRITIC[71]遵循“生成–校验–修正”的路径，模型在产出初始结论后，会主动调用搜索引擎或代码运行环境等外部组件进行一致性核验，并依据验证结果对原始输出进行润色与迭代。

4　挑战与展望

本文将智能原生计算任务生成分为物联网计算意图解析和物联网计算规划生成两个阶段。物联网计算规划生成进一步细分为智能原生计算物联网任务生成和计算任务生成范式。尽管智能原生下物联网计算任务生成已经有一些代表性的研究进展和成果，但整个流程仍存在诸多理论和技术问题亟待解决和完善。本节将讨论智能原生计算任务生成的几个挑战。

1) 如何让大语言模型根据用户需求快速构建软硬件协同的物联网应用?

现有大语言模型主要在软件层面满足用户的应用需求，而能够满足用户需求的物联网软件和硬件具备多样性。不同物联网硬件的价格不同，并且不同硬件完成相关功能所需的功耗不同。因此，如何让大语言模型根据用户需求，自动构建低成本低功耗软硬件协同的物联网应用？利用图结构[72]表达建立软件与硬件的关联关系，根据功能到设备的探索关系，分步获取最优软硬件组合，降低探索开销。并通过组合方式，所有应用抽象为一个实体便于管理，从而让大语言模型更好地理解物联网软硬件信息，满足用户需求。

2) 如何让大语言模型根据用户需求和异构设备特性快速生成AI模型?

虽然现有的大语言模型具备强大的AI应用生成能力，并且能够根据用户需求构建不同的AI模型架构。但是AI模型包含两部分：模型架构和模型参数。现有的大语言模型在生成AI模型架构后，仍需要基于特定的数据集训练AI模型，产生严重的训练开销。那能否让大语言模型根据用户需求和设备计算能力自动生成最优的模型架构和模型参数呢？可以基于一次训练、多种部署（OFA, once for all）架构满足不同设备的计算需求，并让大语言模型学习OFA模型参数训练过程中数据与模型参数变化间的关联关系，通过大语言模型的结果生成过程表征OFA模型训练过程。从而实现让大语言模型根据用户需求和异构设备特性快速生成AI模型。

3) 如何让大语言模型根据物联网计算需求自动生成高质量、可靠的计算任务代码?

虽然现有的大语言模型具备较强的计算任务代码生成能力，能够根据用户需求自动编写物联网计算代码。但生成代码质量包含两部分：代码结构设计和代码实现质量。现有大语言模型在生成代码结构后，仍存在准确性、稳定性不足，且缺乏系统性评估手段，容易产生潜在风险，带来严重的安全隐患。那么，能否让大语言模型在生成过程中就具备对代码质量的自动优化和科学评估能力呢？可以借助“静态分析—动态测试—形式化验证—持续监控”四维评估体系满足物联网设备对代码质量的多维需求，并让大语言模型学习评估过程中代码特征与质量指标之间的关联关系，通过大语言模型的生成过程表征代码质量演化过程。由此，实现大语言模型根据用户需求和设备特性自动生成高质量、合理性强、稳定性优的物联网计算任务代码，为智能原生计算提供可靠保障。

5　结束语

本文针对物联网应用开发背景下，结合大语言模型的快速发展，提出了一种新的编程开发模

式——智能原生计算任务生成。本文首先介绍了传统的物联网应用编程和低代码编程方法，并将其与智能原生计算任务生成进行对比。接着，基于现有的大语言模型的研究进展，本文抽象地构建了智能原生计算任务生成的框架，重点讨论了框架中的各个步骤的研究进展。最后，本文探讨了实现智能原生计算任务生成所面临的关键开放性挑战，并指出了未来可能的研究方向，以促进这一领域的进一步发展。

参考文献：

[1] 吴吉义, 李文娟, 曹健, 等. 智能物联网AIoT研究综述[J]. 电信科学, 2021, 37(8): 1-17.
Wu J Y, Li W J, Cao J, et al. AIoT: a taxonomy, review and future directions[J]. Telecommunications Science, 2021, 37(8): 1-17.

[2] 孙利民, 郭雅, 李瑞, 等. 智能物联网：概念、体系架构与关键技术[J]. 计算机学报, 2023, 46(11): 2259-2278.
Sun L M, Guo Y, Li R, et al. Intelligent Internet of things: concepts, system architecture, and key technologies[J]. Chinese Journal of Computers, 2023, 46(11): 2259-2278.

[3] 袁牧, 张兰, 姚云昊, 等. 面向智能物联网的资源高效模型推理综述[J]. 计算机学报, 2024, 47(10): 2247-2273.
Yuan M, Zhang L, Yao Y H, et al. Resource-efficient model inference for AIoT: a survey[J]. Chinese Journal of Computers, 2024, 47(10): 2247-2273.

[4] 杨震, 曹宁, 张琳, 等. 面向物联网应用的人工智能相关技术研究[J]. 电信技术, 2016(5): 16-19, 23.
Yang Z, Cao N, Zhang L, et al. Research on artificial intelligence-related technologies for Internet of things applications[J]. Telecommunication Technology, 2016(5): 16-19, 23.

[5] Dong W, Li B R, Guan G Y, et al. TinyLink: a holistic system for rapid development of IoT applications[J]. IEEE Transactions on Mobile Computing, 2020, 17(1): 1-29.

[6] Guan G Y, Li B R, Gao Y, et al. TinyLink 2.0: integrating device, cloud, and client development for IoT applications[C]//Proceedings of the 26th Annual International Conference on Mobile Computing and Networking. New York: ACM, 2020: 1-13.

[7] 李博睿, 董玮, 高艺, 等. 物联网软件低代码开发[J]. 中国计算机学会通讯, 2022, 18(11): 18-24.
Li B R, Dong W, Gao Y, et al. Low-code development for Internet of things software[J]. Communications of the Chinese Computer Federation, 2022, 18(11): 18-24.

[8] 苏伟, 国建勋, 冯宽. 低代码开发平台发展现状及标准化研究[J]. 信息技术与标准化, 2024(1): 17-21.
Su W, Guo J X, Feng K. Research on the development status and standardization of low code development platforms[J]. Information Technology & Standardization, 2024(1): 17-21.

[9] Richardson C, Rymer J. New development platforms emerge for customer-facing applications[R]. Cambridge: Forrester Research, 2014.

[10] Gupta T, Kembhavi A. Visual programming: compositional visual reasoning without training[C]//Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE Press, 2023: 14953-14962.

[11] Hu Y H, Yang J Z, Chen L, et al. Planning-oriented autonomous driving[C]//Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE Press, 2023: 17853-17862.

[12] 陈慕涵, 郭佳佳, 李潇, 等. 基于深度学习的大规模MIMO信道状态信息反馈[J]. 物联网学报, 2020, 4(1): 33-44.
Chen M H, Guo J J, Li X, et al. An overview of the CSI feedback based on deep learning for massive MIMO systems[J]. Chinese Journal on Internet of Things, 2020, 4(1): 33-44.

[13] 廖勇, 姚海梅, 花远肖. 一种基于深度学习的物联网信道状态信息获取算法[J]. 物联网学报, 2019, 3(1): 8-13.
Liao Y, Yao H M, Hua Y X. Channel state information acquisition algorithm based on deep learning for IoT[J]. Chinese Journal on Internet of Things, 2019, 3(1): 8-13.

[14] Becker B A, Denny P, Finnie-Ansley J, et al. Programming is hard - or at least it used to be: educational opportunities and challenges of AI code generation[C]//Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1. New York: ACM, 2023: 500-506.

[15] Liu J W, Xia C S, Wang Y Y, et al. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation[C]//Proceedings of the 37th International Conference on Neural Information Processing Systems. New York: ACM, 2023: 21558-21572.

[16] Dong Y H, Jiang X, Jin Z, et al. Self-collaboration code generation via ChatGPT[J]. ACM Transactions on Software Engineering and Methodology, 2024, 33(7): 1-38.

[17] Chang Y P, Wang X, Wang J D, et al. A survey on evaluation of large language models[J]. ACM Transactions on Intelligent Systems and Technology, 2024, 15(3): 1-45.

[18] Zhao W X, Zhou K, Li J, et al. A survey of large language models[J]. AI Open, 2023, 4: 30-52.

[19] Hadi M U, Tashi A, Qureshi R, et al. A survey on large language models: applications, challenges, limitations, and practical usage[J]. TechRxiv (2023-07-10) [2025-06-11]. DOI: 10.36227/techrxiv.23589741.v3.

[20] Cheng Y, Zhang C, Zhang Z, et al. Exploring large language model based intelligent agents: definitions, methods, and prospects[J]. arXiv preprint, 2024, arXiv: 2401.03428.

[21] Xi Z H, Chen W X, Guo X, et al. The rise and potential of large language model based agents: a survey[J]. Science China Information Sciences, 2025, 68(2): 121101.

[22] 闫啸彤, 唐晓彬, 沈童, 等. 大语言模型发展综述[J]. 统计学报, 2024, 5(4): 13-18.
Yan X T, Tang X B, Shen T, et al. Overview of the development of

large language models[J]. Journal of Statistics, 2024, 5(4): 13-18.

[23] Huang X, Liu W, Chen X, et al. Understanding the planning of LLM agents: a survey[PP]. (2024-02-05) [2025-06-11] arXiv: 2402.02710.

[24] 郭先会, 张梦姣, 马军. 基于大语言模型的智能体构建综述[J]. 通信技术, 2024, 57(9): 873-879.
Guo X H, Zhang M J, Ma J. Overview of intelligent agent construction based on large language model[J]. Communications Technology, 2024, 57(9): 873-879.

[25] 王文晟, 谭宁, 黄凯, 等. 基于大模型的具身智能系统综述[J]. 自动化学报, 2025, 51(1): 1-19.
Wang W S, Tan N, Huang K, et al. A survey on embodied intelligent systems based on large models[J]. Acta Automatica Sinica, 2025, 51(1): 1-19.

[26] Zhang D, Yu Y, Dong J, et al. MM-LLMs: recent advances in Multi Modal large language models[PP]. (2024-01-24) [2025-06-11] arXiv: 2401.13601.

[27] Vizniuk A, Diachenko G, Laktionov I, et al. A comprehensive survey of retrieval-augmented large language models for decision making in agriculture: unsolved problems and research opportunities[J]. Journal of Artificial Intelligence and Soft Computing Research, 2024, 15(2): 115-146.

[28] 高一, 蔡瑞初, 李嘉杰, 等. 基于大型语言模型的检索增强生成综述[J]. 计算机工程与应用, 2024, 60(14): 1-17.
Gao Y, Cai R C, Li J J, et al. A survey on retrieval-augmented generation based on large language models[J]. Computer Engineering and Applications, 2024, 60(14): 1-17.

[29] Fan X X, Deng X J, Xia Y Z, et al. Tensor-based confident information coverage reliability of hybrid Internet of Things[J]. IEEE Transactions on Mobile Computing, 2024, 23(3): 2171-2185.

[30] Gong W, Liu K B, Liu Y H. Directional diagnosis for wireless sensor networks[J]. IEEE Transactions on Parallel and Distributed Systems, 2015, 26(5): 1290-1300.

[31] Deng X J, Wang B, Liu W Y, et al. Sensor scheduling for multi-modal confident information coverage in sensor networks[J]. IEEE Transactions on Parallel and Distributed Systems, 2015, 26(3): 902-913.

[32] Liu K B, Ma Q, Gong W, et al. Self-diagnosis for detecting system failures in large-scale wireless sensor networks[J]. IEEE Transactions on Wireless Communications, 2014, 13(10): 5535-5545.

[33] Sahay A, Indamutsa A, Di Ruscio D, et al. Supporting the understanding and comparison of low-code development platforms[C]// Proceedings of the 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). Piscataway: IEEE Press, 2020: 171-178.

[34] 董玮, 董健, 郭艺. 物联网低代码开发平台及应用[J]. 计算机测量与控制, 2021, 29(12): 253.
Dong W, Dong J, Guo Y. Low-code development platform and applications for the Internet of things[J]. Computer Measurement & Control, 2021, 29(12): 253.

[35] Luo Y J, Liang P, Wang C, et al. Characteristics and challenges of low-code development: the practitioners' perspective[C]//Proceedings of the 15th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). New York: ACM Press, 2021: 1-11.

[36] 周彤. 基于端云一体化的可重构物联网系统平台设计[D]. 杭州: 杭州电子科技大学, 2022.
Zhou T. Design of reconfigurable Internet of things system platform based on edge-cloud integration[D]. Hangzhou: Hangzhou University of Electronic Science and Technology, 2022.

[37] Li B R, Dong W. EdgeProg: edge-centric programming for IoT applications[C]//Proceedings of the 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS). Piscataway: IEEE Press, 2020: 212-222.

[38] Li B R, Dong W. Automatic generation of IoT device platforms with AutoLink[J]. IEEE Internet of Things Journal, 2021, 8(7): 5893-5903.

[39] Gong W, Stojmenovic I, Nayak A, et al. Fast and scalable counterfeits estimation for large-scale RFID systems[J]. IEEE/ACM Transactions on Networking, 2016, 24(2): 1052-1064.

[40] Wang Q W, Chen S, Zhao J, et al. RapidRider: efficient WiFi backscatter with uncontrolled ambient signals[C]//Proceedings of the IEEE INFOCOM 2021 - IEEE Conference on Computer Communications. Piscataway: IEEE Press, 2021: 1-10.

[41] Wang G, Han L B, Chang Y C, et al. Cross-technology communication between visible light and battery-free RFIDs[J]. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2023, 7(3): 1-20.

[42] Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16x16 words: transformers for image recognition at scale[C]//Proceedings of the 9th International Conference on Learning Representations (ICLR). Amherst: ICLR, 2021: 1-21.

[43] Chen S, Wu Y, Wang C, et al. BEATs: audio pre-training with acoustic tokenizers[C]//Proceedings of the 40th International Conference on Machine Learning (ICML). New York: ACM Press, 2023: 1-15.

[44] Chen F, Han M, Zhao H, et al. X-LLM: bootstrapping advanced large language models by treating multi-modalities as foreign languages[PP]. (2023-05-07) [2025-06-11]. arXiv: 2305.04160.

[45] Yuan J L, Yang C, Cai D Q, et al. Mobile foundation model as firmware[C]//Proceedings of the 30th Annual International Conference on Mobile Computing and Networking. New York: ACM Press, 2024: 279-295.

[46] Huang K, Yang B, Gao W. Modality plug-and-play: elastic modality adaptation in multimodal LLMs for embodied AI[C]//Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE Press, 2023: 14935-14944.

[47] Suris D, Menon S, Vondrick C. ViperGPT: visual inference via Python execution for reasoning[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). Piscataway:

IEEE Press, 2023: 11588-11598.

[48] Liu Z, He Y, Wang W, et al. InternGPT: solving vision-centric tasks by interacting with ChatGPT beyond language[PP]. (2023-05-09) [2025-06-11]. arXiv: 2305.05662.

[49] Shen Y, Song K, Tan X, et al. HuggingGPT: solving AI tasks with ChatGPT and its friends in hugging face[C]//Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS). Red Hook: Curran Associates, 2023: 1-13.

[50] Qian C, Liu W, Liu H, et al. ChatDev: communicative agents for software development[C]//Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL). Stroudsburg: ACL, 2024: 1-12.

[51] Hong S, Zhuge M, Chen J, et al. MetaGPT: meta programming for a multi-agent collaborative framework[C]//Proceedings of the 12th International Conference on Learning Representations (ICLR). Amherst: ICLR, 2024: 1-24.

[52] Shen L M, Zheng Y Q. IoTCoder: a copilot for IoT application development[C]//Proceedings of the 30th Annual International Conference on Mobile Computing and Networking. New York: ACM Press, 2024: 1647-1649.

[53] Gong K J, Dong W, Peng Y Q, et al. Poster: enabling IoT application programming in natural language with IoTPilot[C]//Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems. New York: ACM Press, 2024: 903-904.

[54] Yang H Q, Li M Z, Han M D, et al. EmbedGenius: towards automated software development for generic embedded IoT systems[PP]. (2024-12-04) [2025-06-11]. arXiv: 2412.09058.

[55] Cheng Y, Xu M H, Zhang Y, et al. AutoIoT: automated IoT platform using large language models[J]. IEEE Internet of Things Journal, 2025, 12(10): 13644-13656.

[56] Gao Y, Xiao K J, Li F, et al. ChatIoT: zero-code generation of trigger-action based IoT programs[J]. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2024, 8(3): 1-29.

[57] Hu C, Fu J, Du C, et al. ChatDB: augmenting LLMs with databases as their symbolic memory[PP]. (2023-06-06) [2025-06-11]. arXiv: 2306.03901.

[58] Zhou X, Sun J, Zhang G, et al. DB-GPT: large language model meets database[PP]. (2023-06-24) [2025-06-11]. arXiv: 2306.14101.

[59] Ren Y, Yu C, Li W, et al. TableGPT: a novel table understanding method based on table recognition and large language model collaborative enhancement[J]. Applied Intelligence, 2025, 55(5): 311.

[60] King E, Yu H X, Lee S, et al. Sasha: creative goal-oriented reasoning in smart homes with large language models[J]. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2024, 8(1): 1-38.

[61] Rivkin D, Hogan F, Feriani A, et al. AIoT smart home via autonomous LLM agents[J]. IEEE Internet of Things Journal, 2025, 12(3): 2458-2472.

[62] Brown T, Mann B, Ryder N, et al. Language models are few-shot learners[C]//Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS). Red Hook: Curran Associates, 2020: 1877-1901.

[63] Wei J, Wang X Z, Schuurmans D, et al. Chain-of-thought prompting elicits reasoning in large language models[C]//Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS). Red Hook: Curran Associates, 2022: 24824-24837.

[64] Kojima T, Gu S S, Reid M, et al. Large language models are zero-shot reasoners[C]//Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS). Red Hook: Curran Associates, 2022: 22199-22213.

[65] Wang X, Wei J, Schuurmans D, et al. Self-consistency improves chain of thought reasoning in language models[C]//Proceedings of the 11th International Conference on Learning Representations (ICLR). Amherst: ICLR, 2023: 1-15.

[66] Yao S, Yu D, Zhao J, et al. Tree of thoughts: deliberate problem solving with large language models[C]//Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS). Red Hook: Curran Associates, 2023: 11809-11822.

[67] Yao S, Zhao J, Yu D, et al. ReAct: synergizing reasoning and acting in language models[C]//Proceedings of the 11th International Conference on Learning Representations (ICLR). Amherst: ICLR, 2023: 1-10.

[68] Madaan A, Tandon N, Gupta P, et al. Self-refine: iterative refinement with self-feedback[C]//Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS). Red Hook: Curran Associates, 2023: 46534-46594.

[69] Zhou D, Schärli N, Hou L, et al. Least-to-most prompting enables complex reasoning in large language models[C]//Proceedings of the 11th International Conference on Learning Representations (ICLR). Amherst: ICLR, 2023: 1-12.

[70] Shinn N, Cassano F, Gopinath A, et al. Reflexion: language agents with verbal reinforcement learning[C]//Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS). Red Hook: Curran Associates, 2023: 8634-8652.

[71] Gou Z, Shao Z, Gong Y, et al. CRITIC: large language models can self-correct with tool-interactive critiquing[C]//Proceedings of the 12th International Conference on Learning Representations (ICLR). Amherst: ICLR, 2024: 1-18.

[72] 王浩彬, 皇甫伟, 刘娅汐, 等. 基于计算图的移动通信网络物联网业务覆盖优化算法及实现[J]. 物联网学报, 2019, 3(2): 100-107.
Wang H B, Huang F W, Liu Y X, et al. Coverage optimization algorithm and implementation based on computational graph for mobile communication network and IoT service[J]. Chinese Journal on Internet of Things, 2019, 3(2): 100-107.

[作者简介]

束方磊（2000– ），男，东南大学计算机科学与工程学院硕士生，主要研究方向为大语言模型智能体、边缘计算。

李博睿（1994– ），男，博士，东南大学计算机科学与工程学院助理教授，主要研究方向为边缘计算、智能物联网。

刘佳伟（1996– ），男，东南大学计算机科学与工程学院博士生，主要研究方向为具身智能、智能穿戴、深度学习、模式识别。

王帅（1987– ），男，博士，东南大学计算机科学与工程学院青年首席教授，主要研究方向为物联网、大数据分析、无线通信。