Computer Science and Application 计算机科学与应用, 2023, 13(7), 1492-1506
Published Online July 2023 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2023.137148

基于组件化和M2M的物联网系统开发范式研究

邱林山[1]，程良伦[1,2]

[1]广东工业大学自动化学院，广东 广州
[2]广东工业大学计算机学院，广东 广州

收稿日期：2023年6月24日；录用日期：2023年7月21日；发布日期：2023年7月28日

摘 要

随着物联网技术的不断发展，涌现了越来越多的物联网应用系统。物联网愿景是将尽可能多的设备变成屏蔽底层差异的服务，和互联网服务进行无缝的集成，为广大人民提供便利。但是，目前物联网应用大多采用封闭紧耦合的竖直型的范式进行开发，使用文本化编程的方式进行开发。这就带来了诸多的不便：开发门槛高、系统可维护性差、不便于跨平台、不便于设备集成。为此本文提出了组件化开发的模式，定义了组件模型，提出了基于组件化和M2M的物联网系统开发范式。本文提出的软件组件模型可以屏蔽系统开发过程中存在的平台、操作系统、编程语言、网络环境等高度异构的问题，实现应用程序的“Write Once, Run Anywhere”的能力。建立了组件化应用软件应用的模型，使用DAG图进行了形式化描述和验证，由本文提出的组件模型构成的应用程序符合有向无环图。提出了基于组件化的M2M的物联网系统开发范式，为高效快速的搭建跨平台、跨操作系统的物联网系统提供了可行的参考。根据组件模型，开发了数据采集组件、数据处理组件、数据展示组件，这些组件是可以跨平台的，实现了“Write Once, Run Anywhere”的特点，搭建了组件化的智能家居系统，验证了组件模型的有效性，所搭建组件的可用性，在基于Linux系统的树莓派和Windows系统的笔记本上运行组件的物联网系统，验证了组件的跨平台性，以及验证了本文组件的M2M的物联网开发范式的可用性。

关键词

组件模型，组件化，M2M，物联网

Research on Development Paradigm of Internet of Things System Based on Componentization and M2M

Linshan Qiu[1], Lianglun Cheng[1,2]

[1]School of Automation, Guangdong University of Technology, Guangzhou Guangdong
[2]School of Computer Science and Technology, Guangdong University of Technology, Guangzhou Guangdong

Received: Jun. 24th, 2023; accepted: Jul. 21st, 2023; published: Jul. 28th, 2023

文章引用: 邱林山, 程良伦. 基于组件化和 M2M 的物联网系统开发范式研究[J]. 计算机科学与应用, 2023, 13(7): 1492-1506. DOI: 10.12677/csa.2023.137148

Abstract

With the continuous development of Internet of Things (IoT) technology, more and more IoT application systems have emerged. The vision of the Internet of Things is to turn as many devices as possible into services that shield the underlying differences, and seamlessly integrate with Internet services to provide convenience for the majority of people. However, at present, most of the IoT applications are developed in a closed and tightly coupled vertical paradigm, and are developed by textual programming. This brings a lot of inconveniences: high development threshold, poor system maintainability, not convenient for cross-platform, and not convenient for equipment integration. To this end, this paper proposes the pattern of component-based development, defines the component model, and proposes the component-based and M2M based IoT system development paradigm. A component-based IoT system is built to verify the usability of the component model and development paradigm. The software component model proposed in this paper can abstract the highly heterogeneous issues in system development, such as platforms, operating systems, programming languages, and network environments, enabling the "Write Once, Run Anywhere" capability of applications. A model for component-based application software is established, and a formal description and validation using a Directed Acyclic Graph (DAG) are performed. The application programs constructed using the proposed component model conform to the properties of a Directed Acyclic Graph. A component-based paradigm for M2M-based IoT system development is proposed, providing a feasible reference for efficiently and rapidly building cross-platform, cross-operating system IoT systems. Based on the component model, data acquisition components, data processing components, and data display components are developed, which are platform-independent and exhibit the "Write Once, Run Anywhere" feature. A componentized smart home system is built to validate the effectiveness of the component model and the usability of the constructed components. The IoT system running the components is tested on both a Raspberry Pi based on the Linux system and a laptop running the Windows system to verify the cross-platform nature of the components, as well as to validate the usability of the component-based M2M IoT development paradigm proposed in this paper.

Keywords

Component Model, Componentization, M2M, Internet of Things

1. 背景介绍

物联网可以将传感器和设备连接起来，从而实现自动化控制、监测和维护等功能[1]。此外，物联网还可以应用于智能家居、智能城市、智能医疗等领域。物联网已经成为广受关注的研究领域，其旨在将物理世界中的所有物体连接到互联网中，实现智能化的互联通信，从而达到人、机器和物体之间的无缝沟通，最终构建一个全面智能化的和谐世界。物联网利用互联网连接物理世界，为全球范围内的用户提供智能化的服务。

目前大多数的物联网应用是采用文本化编程和竖直型的开发方法，随着物联网设备指数级的增加以及物联网应用的逐渐普及，这种开发方法开始显现出很多缺点。

在物联网系统的开发中，大多采用文本化编程的方法开发，这种方法具有以下缺点：

1) 技术门槛高

传统的文本化编程需要掌握一定的编程语言、语法和算法等专业知识，学习成本较高，需要花费大量时间和精力进行学习和实践。

2) 可读性差

由于传统的文本化编程使用的是纯文本形式的代码，代码的可读性较差，不够直观和易于理解。在处理复杂的程序逻辑和数据结构时，需要花费更多的时间和精力。

3) 可维护性差

传统的文本化编程往往需要编写大量的代码，程序的维护成本较高。如果程序的结构和逻辑不够清晰和简洁，就会给后续的维护和修改带来困难。

4) 不便于跨平台

由于不同操作系统和编程环境的差异，传统的文本化编程往往需要编写不同的代码来适应不同的平台，这会增加开发和维护的难度和成本。

5) 缺乏交互性

传统的文本化编程方式缺乏直观的交互性，不能够直接显示程序的执行结果和运行状态，需要通过编写额外的代码来实现。这会增加程序的开发难度和调试时间。

现有的竖直型物联网开发模式的缺点：

缺乏通用性：竖直型的物联网开发模式通常面向特定的行业或竖直市场，因此其解决方案可能缺乏通用性。这意味着相同的解决方案不一定适用于其他行业或市场，限制了其在跨行业和跨市场应用的能力。

开发成本高：由于竖直型物联网开发模式需要深入了解特定行业和需求，以及定制化解决方案的设计和开发，因此开发成本可能较高。这包括专业人才的招聘和培训、定制硬件和软件的开发等。

市场限制：竖直型物联网开发模式在特定行业或竖直市场内运作，其市场规模和增长潜力可能受到限制。这可能导致竞争加剧和市场份额有限的问题。

在物联网中，不同类型的设备和信息生成方式多种多样，这些设备的接入具有很强的异构性。因此，对于不同领域的应用需求，感知数据的信息融合变得至关重要。这样，就能够实现大规模、异构的感知节点之间的互联和互通。目前，物联网应用开发主要采用竖直方式进行，但这种方式存在碎片化的问题[2]。因此，如何促进不同设备和应用系统之间的数据交互共享，实现 IoT 实体设备的服务化，协调 IoT 组件的支持以及开发跨领域和跨平台的物联网应用，成为物联网发展的一个重要问题。

针对以上问题，本文提出组件化的 M2M 物联网开发方法，设计了基于流的组件模型，提出了基于组件和 M2M 技术的物联网应用开发方法。

2. 研究现状

组件技术，旨在将大型复杂的软件应用程序分解成多个独立的、松耦合的、可扩展的软件单元，即组件[3]。每个组件可以实现不同的功能，可通过统一的接口进行交互，从而提高了系统的可维护性和可扩展性。这些组件可被组合在一起，形成高度可复用的软件应用程序。组件技术具有良好的可维护性和可重用性，使得开发人员可以更快速地构建出高质量的软件应用程序[4]。在组件技术中，组件被看作是一种独立的、可移植的、可替换的、可扩展的软件实体，因此，它们可以轻松地被替换、更新或删除。这种灵活性为软件应用程序的开发和维护带来了便利，也为软件行业带来了更广阔的发展空间[5]。总之，组件技术是软件工程领域中非常重要的一种技术手段，可以实现工业软件的快速开发，同时也能够促进软件行业的发展。

组件技术具有高效和经济的优势，得到了更好的支持和实践平台。随着时间的推移，组件的概念从最初的子程序逐渐扩大为系统级抽象，并不断成熟和发展[6]。1995 年左右，随着面向对象技术的进步，组件化应用开始得到越来越广泛的应用。组件化应用的优势在于提高了软件开发的效率，同时也降低了成本。组件技术的广泛应用促进了它的不断发展和完善，并为软件行业带来了更多的机遇和挑战[7]。面向对象技术基于类的组合，利用封装隐藏实现细节，提供服务接口，以便于外部服务请求和提供服务环境。随着计算机技术的发展，应用软件变得越来越庞大和复杂。为了更好地管理和开发这些系统，软件开发人员开始采用分层思想和组件化的方法来实现模块化开发，将整个系统拆分成多个不同的层次和模块，以提高效率。这种方法将大而复杂的软件分成多个可以独立开发的软件单元，而且也是一种分层的系统设计方法，将系统分割成可控的、易于维护的模块。

对于组件规范化，已经有了有一些研究成果。表 1 列出了目前的一些组件模型。这些规范包括接口规范、交互规范、生命周期规范等，这些规范有助于开发人员在设计和实现组件时遵循一致的标准和最佳实践。

Table 1. Organizations that develops component specifications
表 1. 制定组件规范的机构

机构	组件模型
OMG (Object Management Group)组织	CORBA (Common Object Request Broker Architecture)模型 DDS (Data Distribution Service)模型
Microsoft 公司	COM/DCOM (Component Object Model/Distributed Component Object Model)模型
Sun Microsystem 公司	EJB (Enterprise JavaBean)模型

然而，目前还没统一的组件模型，不同企业的组件开发和组装平台是不一样的，导致无法统一描述组件的形式。这种情况使得组件不能够跨平台，跨系统，给软件开发和应用带来了不便[8]。

2020 年 Nora 等人开发了 DeepScratch，DeepScratch 是一个面向深度学习应用开发的组件化编程平台，它基于 Scratch 的模块化编程思想，使得深度学习模型的开发过程更加简单和直观。DeepScratch 提供了大量的可视化组件，包括数据预处理、模型构建、训练和评估等模块，用户可以通过拖放组件的方式快速搭建深度学习模型，并通过组件化界面进行调试和优化[9]。此外，DeepScratch 还支持多种深度学习框架，例如 TensorFlow 和 PyTorch，用户可以选择自己熟悉的框架进行开发。由于 DeepScratch 的组件化编程思想和可视化界面，使得深度学习应用的开发过程更加易于理解和上手，适合初学者和专业开发人员使用。

汤世征等人于 2020 年提出了一种基于百度 Paddle 框架的组件化编程系统，旨在简化深度学习模型的构建过程，并通过异步任务调度和可视化操作面板方便用户查看任务执行状态和展示数据。该系统支持重用经典模型和预训练模型。用户只需在本地部署环境和应用程序即可使用[10]。

国内外研究者已经对组件化取得了良好的研究成果。然而，对于软件组件化，相关研究还不够充分，而且在交互性、易用性、跨平台性等方面还存在一些问题。因此，对于软件组件化研究工作还有待深入展开。

近年来，M2M 通信作为一项新型通信技术，普遍应用于人们的日常生产生活中[11]。M2M 通信通常是指在没有或少量人工干预的情况下，机器类设备(Machine Type Communication Devices, MTCD)通过有线或无线链路进行低功耗、低成本的自主信息传输[12]。常见的 M2M 通信系统是由传感器、网关、无线或有线网络、服务器等设备组成[13]，其中传感器负责采集数据，网关负责控制设备之间或 M2M 网络与其他网络之间的连接，网络负责承载 M2M 设备与终端之间的信息传输，服务器或其他终端负责存储、计算或处

理 MTCD 上传的数据[14] [15]。目前 M2M 已经应用于许多行业中，包括工业自动化、远程信息处理、电子医疗、远程监控、公共安全、智能交通、车队管理、电子消费、智能电网和智能办公等领域[16]。

随着 M2M 通信在众多行业中的飞速发展，大量具有自主通信能力的机器类设备开始接入网络、逐步构建了物联网并成为推动其技术进步的主力军之一[17]。

3. 组件模型

目前的文本化开发缺少统一的软件组件模型，无法实现应用程序的“Write Once, Run Anywhere”的能力。本文设计一种可重用的模块给用户使用，也就是一种软件组件模型，能够显著减少相应工作人员的工作量。在开发新的物联网应用时，可以使用先前开发的模块，提高开发效率。一个功能模块就是一个组件。开发人员还可以共享自己的组件，避免工作人员“重复地造轮子”。组件化编程可以降低编程门槛和难度，让使用者可以更专注于应用程序的逻辑，而不必关心具体的编码实现、框架选择和语法问题。

在物联网应用开发中，组件是一个既定概念，能够有效地用于定义鲁棒性、可复用的软件组件。组件允许将工业算法封装为非软件专家也能够轻易理解和使用的形式，并通过提高抽象度以应对系统设计的复杂性。组件也有自己的数据结构，是一种可以集成或内嵌算法进行操作的软件功能单元，每一个组件都包含外部端口和内部功能。

组件内部算法的执行，都需要有数据的输入才可以触发执行，在算法执行完毕后会把结果输出到输出端口。根据高内聚的特点，良好设计的组件不需要对外披露太多的内部细节。在整个应用中，不存在全局变量，数据只可以通过组件进行传递。根据以上原则，本发明设计如图 1 所示的软件组件模型。

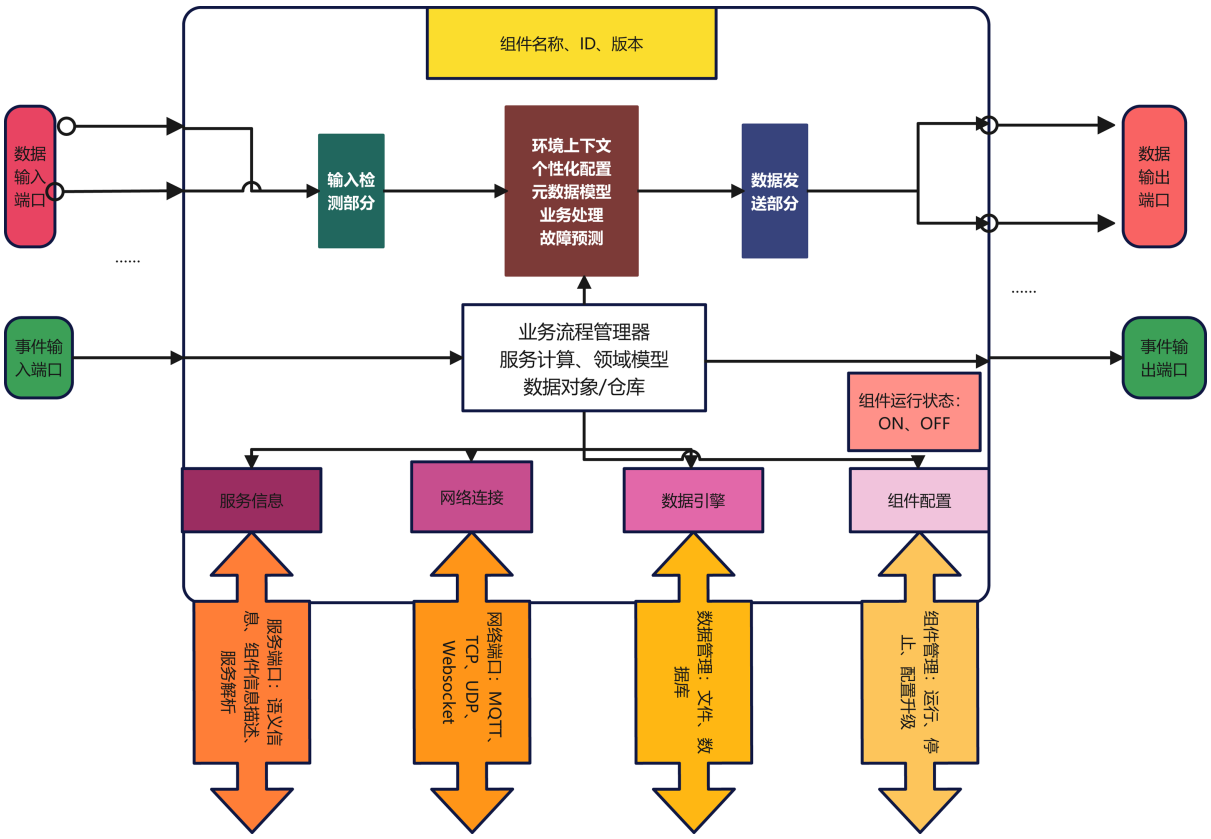

Figure 1. Software component model
图 1. 软件组件模型

借助面向对象编程的思想，每个组件封装一个功能，这样可以做到高内聚、低耦合。每个组件，类似于 Java 语言中的类(Class)，组件实例类似于 Java 中的实例(Instance)。

组件的命名规则：标识符的第一个字符不能是数字；标识符不能含有空格字符，也不能使用两个及以上的连续下划线字符。

组件的 ID 是唯一标识，类似与每个人的身份证号码。组件名字可以一样，但组件的 ID 是不可以一样的。

3.1. 数据和事件端口

对于每个组件而言，数据和事件是相互绑定的，也即数据和事件是同时到来的。每个组件可以有多个数据端口和事件端口。

每个组件的外部接口包含以下内容：

组件的名字和实例的名字。

组件数据的输入和输出的数量、名字、初始值、数据类型，顺序。

所有文本信息只能使用 ISO/IEC646：1991 表 2 规定的字符。

标识符的第一个字符不能是数字。

标识符不能含有空格字符，也不能使用两个及以上的连续下划线字符。所以，MUL，_MUL，M_U_L，等是合乎规范的名字。而 1MUL，M___UL，MUL，等是不合规范的名字。

组件的输入和输出内容可以是基本数据类型，也可以是 JSON、XML 等。

如图 1 所示，数据输入接口、数据输出接口、事件输入接口、事件输出接口、组件服务信息接口、组件网络连接接口、组件数据引擎接口、组件配置管理接口。组件可以分为：有输出和输入的组件、无输入有输出的组件。不存在有输入但无输出的组件。

每个组件在被实例化时，其所有的输入、输出及内部变量都会被实例化。在数据被输入之后，会激活组件内部的算法，完成算法后，会有相应的输出。如果有多个数据输入接口，则只有在所有数据都被输入后才可以激活内部封装的算法。

3.2. 服务端口

服务端口可以将自己的信息注册到服务中心，服务中心可以根据服务需求者的要求进行服务组合，可以自动化的构建软件应用。端口的作用非常重要，它同时还承担着以下几个方面的职责：

1) 服务提供者和服务消费者之间的通信

服务端口是服务提供者和服务消费者之间通信的重要媒介。服务提供者通过定义服务端口来公开自己的服务，服务消费者可以通过调用服务端口来访问服务提供者的功能。

2) 隔离和解耦服务实现

服务端口可以隔离和解耦服务实现和服务消费者，使得服务提供者可以独立地开发、测试和部署服务实现，而服务消费者则只需要关注服务端口的定义和使用，不需要了解服务实现的具体细节。

3) 促进服务的灵活性和可扩展性

通过定义服务端口，服务提供者可以对服务实现进行灵活的替换和升级，而不会影响服务消费者的使用。同时，服务提供者也可以根据需要增加新的服务端口来扩展自己的服务功能，以满足不同的业务需求。

4) 提高服务的可维护性和可测试性

通过定义服务端口，服务提供者可以将服务实现分解成多个独立的模块，便于单元测试和模块化开

发。同时，服务端口也可以提高服务的可维护性，使得服务实现的修改和调试更加容易。

总之，服务端口对于融合微服务架构的组件模型的重要组成部分，它可以提高服务的灵活性、可扩展性、可维护性和可测试性，从而使得组件更加具有弹性和敏捷性。

3.3. 网络端口

网络端口可以实现组件的网络通信，每个组件可以实现一种常见的通信协议或多种通信协议。以下是一些常见的网络连接通信协议：

HTTP：用于发送 HTTP 请求和接收 HTTP 响应，可以实现与 Web API 的交互。

TCP：用于建立 TCP 连接并发送数据，支持异步或同步方式，可以实现基于 TCP 协议的通信。

UDP：用于建立 UDP 连接并发送数据，支持异步或同步方式，可以实现基于 UDP 协议的通信。

MQTT：用于实现 MQTT (Message Queuing Telemetry Transport)协议的通信，支持发布和订阅消息。

WebSocket：用于实现 WebSocket 协议的通信，支持双向通信和长连接。

Web Socket Input：用于接收 WebSocket 协议的数据，支持双向通信和长连接。

Email：用于发送电子邮件，支持 SMTP 和 IMAP 协议。

这些端口可以帮助组件实现不同类型的网络连接，从而构建更加强大和灵活的应用程序。例如，使用 HTTP Request 可以方便地与 Web API 进行交互，使用 TCP 和 UDP 可以实现自定义的网络协议，使用 MQTT 可以实现物联网设备之间的通信。

3.4. 数据管理端口

数据管理端口是一种用于处理和管理数据的端口，可以提供数据的存储、查询、分析和处理等服务。数据管理端口通常包括多个组件，如数据库管理系统、数据仓库、数据集成工具、数据分析工具等，可以支持多种数据源和数据格式，如关系型数据库、数据云平台等。

管理端口的主要作用是帮助用户有效地管理和利用数据。具体来说，管理端口可以实现以下功能：

数据存储：提供数据的持久化存储，并支持数据的增删改查操作。

数据集成：将多个数据源的数据整合在一起，并进行转换和映射，以满足不同的业务需求。

数据分析：提供数据分析工具和算法，支持数据的聚合、统计、挖掘等操作，以发现数据中的规律和趋势。

数据可视化：将数据以图表形式展示，使用户可以直观地了解数据的情况和趋势。

数据安全：保护数据的隐私和安全，并控制数据的访问权限，以保证数据的安全性和完整性。

数据管理端口可以在多个领域发挥作用，如企业资源管理、客户关系管理、电子商务、物联网、金融等。它可以帮助企业更好地管理和利用数据，提高业务的效率和竞争力。

3.5. 组件管理配置端口

管理端口的作用主要有以下几点：

管理组件的创建和编辑：用于创建和编辑功能块，管理组件的界面直观友好，可以方便地定义输入输出接口、内部变量、算法逻辑等。

管理组件的复用：一个组件可以被多个应用程序或系统使用，因此可以实现功能块的复用。管理功能块可以方便地查找、复制和粘贴已有的功能块，减少了重复编写代码的工作量。

管理组件的版本和发布：在组件中，组件的版本是可以进行管理的，可以对功能块进行版本控制，方便进行升级和维护。此外，管理端口也可以进行发布，供其他用户或系统使用。

管理组件的测试和调试：在组件中，管理端口可以进行测试和调试，以确保其正确性和稳定性。组件测试可以在不同的环境下进行，例如仿真环境、实验室测试环境等。

管理组件的文档和注释：管理端口可以进行文档和注释管理，方便其他开发人员了解和使用该功能块。

总之，管理端口是组件中非常重要的一个端口，它可以方便地创建、管理、复用和测试功能块。

4. 组件化软件应用程序模型

本文提出了基于 DAG (Directed Acyclic Graph，有向无环图)组件化软件应用程序模型，如图 2 所示。在此应用模型中，应用由多个组件组成，每个应用程序构成一个应用。应用间可以通过网络进行通信。组件可以是用户自定义的可重用组件，也可以是组件库中的标准通用组件。每个组件都封装了某种类型的功能，以实现特定的功能[18]。环境数据采集组件的主要功能是采集环境数据，数据处理组件对采集的环境数据进行分析处理，而数据展示组件则主要负责展示处理后的数据页面。组件之间可以进行数据传输，并可以根据程序需要动态地关联。组件有两种状态：休眠状态和活动状态。整个应用程序的运行过程依赖于数据流来驱动。多个组件就可以并行运行，该软件应用程序模型提供了多线程。

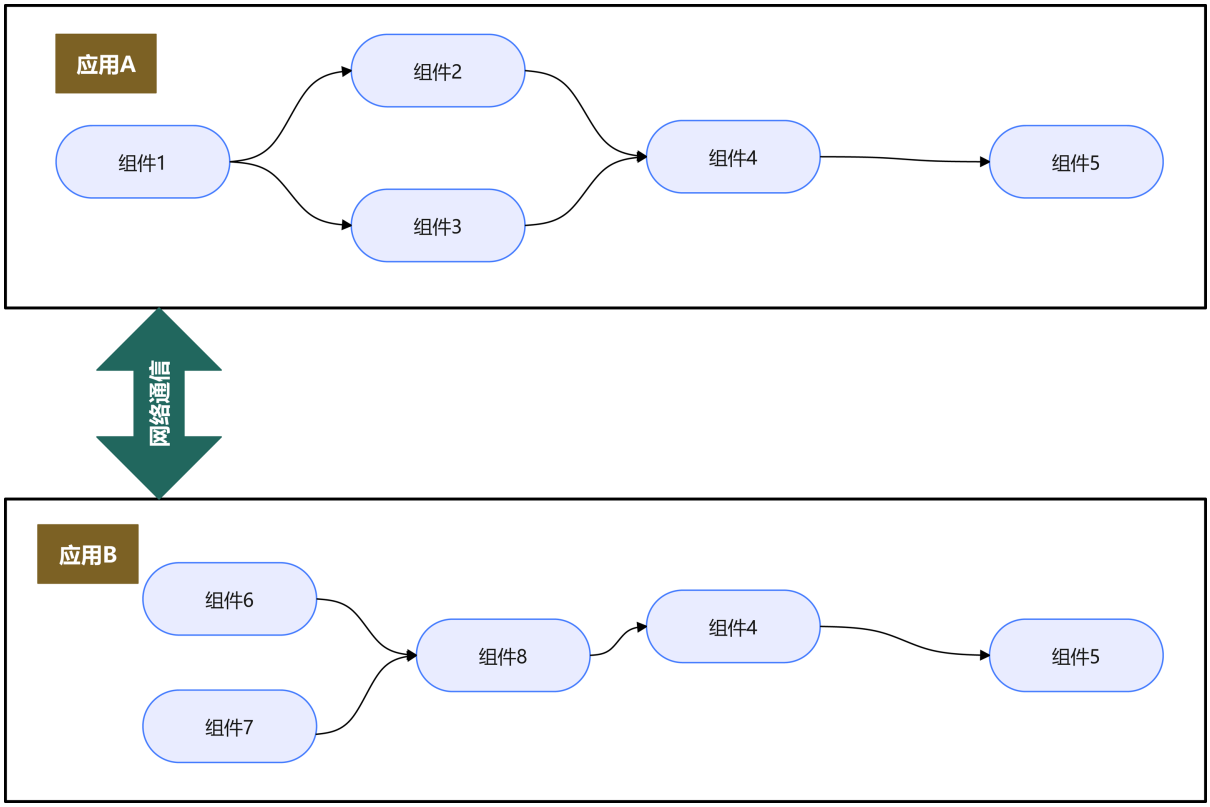

Figure 2. Componentization software application model based on DAG
图 2. 基于 DAG 的组件化软件应用程序模型

组件化编程的应用模型具有简单性、可并行性和易于理解性的特点。这种应用模型可以被视为一个有向图，其中每个组件负责处理特定的数据和逻辑，而连线则负责传输数据。在程序的执行过程中，数据驱动了整个过程，只有在当前节点所需的所有数据都满足条件时，相应的代码才会被执行。这种数据驱动的方式可以帮助开发人员更好地理解程序执行过程中的逻辑，同时也可以提高程序的并行性，从而提高程序的性能。

5. 基于组件和 M2M 技术的物联网系统构建方法

当今数字时代的一个重要概念是“M2M”(machine to machine communication，机器对机器)和“物联网”(Internet of Things)。M2M 指的是通过无线或有线网络连接的机器之间的直接通信，而物联网则是指通过互联网连接和共享信息的各种设备和物体的网络。M2M (Machine-to-Machine)通信是物联网系统中关键的通信方式，通过无线或有线连接实现设备之间的实时数据交换和命令交互。它建立在物联网设备的集成和协作基础上，利用通信技术和协议，以实现高度自动化和智能化的信息交流。M2M 通信的基本原理是通过设备间的数据传输和协议交互实现信息的收集、传递和处理。它依赖于传感器、执行器和其他物联网设备的集成和连接，以获取环境数据、监控设备状态，并通过网络实现远程控制和管理。

在 M2M 通信中，设备通过感知环境中的物理现象、采集传感器数据，并使用通信协议进行数据交换。通常使用的通信技术包括无线网络(如 Wi-Fi、蓝牙、LoRa 等)和有线网络(如以太网、RS-485 等)。通过这些通信技术，设备可以相互传递数据、命令和状态信息。M2M 通信的概念涵盖了通信架构、协议、安全性和可靠性等方面。它需要可靠的通信链路、适当的通信协议和数据格式，以确保设备之间的有效通信和数据交换。同时，M2M 通信还需要考虑数据的安全性和隐私保护，通过加密和身份验证等手段，保证通信过程的安全性。

M2M 和物联网的兴起对我们的生活和商业领域产生了深远的影响。通过 M2M 技术，各种设备和机器可以自动交换数据，进行实时监测和控制。总之，M2M 通信作为物联网系统的核心组成部分，通过设备之间的实时数据传输和协议交互，实现设备的集成、控制和智能化。它为物联网系统提供了高效、可靠和安全的通信基础，推动了物联网技术的发展和应用。

M2M (Machine-to-Machine)通信在物联网系统中扮演着关键的角色，并带来了多项功能和优势，如下所述：

实时数据传输：M2M 通信实现了物联网设备之间的快速、实时的数据传输，使得环境监测、设备控制等应用能够及时获取和响应数据。通过高速的数据传输，物联网系统能够实现实时监测、预警和决策，提升整体效率和响应能力。

远程监测与控制：M2M 通信使得远程监测和控制成为可能，设备可以通过网络进行远程访问和管理。这为物联网系统提供了远程设备监测、故障诊断和远程配置等功能。运维人员可以远程监视设备状态、进行故障排查和远程维护，提高了设备的可用性和系统的可靠性。

自主决策和自适应性：M2M 通信使得物联网设备能够基于收集到的数据进行自主决策和自适应调整。设备可以通过内置的算法和智能决策机制，根据实时数据分析和模型预测，自动调整工作参数、执行任务，并实现对环境变化的自适应响应。这为物联网系统带来了更高的智能化和自动化水平。

优化资源利用：M2M 通信实现了设备之间的协同工作和资源优化利用。物联网系统中的设备可以通过 M2M 通信共享信息、协调任务和资源分配，提高能源利用效率、减少资源浪费。例如，基于 M2M 通信的智能能源管理系统可以实时监测能源消耗，通过设备间的协作和调节，优化能源分配和节约能源成本。

扩展性和互操作性：M2M 通信为物联网系统提供了高度的扩展性和互操作性。通过统一的通信协议和标准化的接口，不同厂商和设备之间可以实现互联互通，无缝集成到物联网系统中。这使得物联网系统的扩展和升级更加灵活，可以根据需求添加新的设备或替换现有设备，而无需对整个系统进行重构。

本文提出的基于组件化和 M2M 技术的物联网应用程序架构如图 3 所示，边缘端的传感器和执行器等边缘端设备连接到边缘计算设备，终端应用端设备访问应用服务器可以得到边缘端设备的数据和状态，从而能够控制边缘端的设备。边缘计算设备和应用服务器之间通过网关和 MQTT 代理进行数据传输，边缘端和应用服务器之间通过发布/订阅模式进行数据传输。

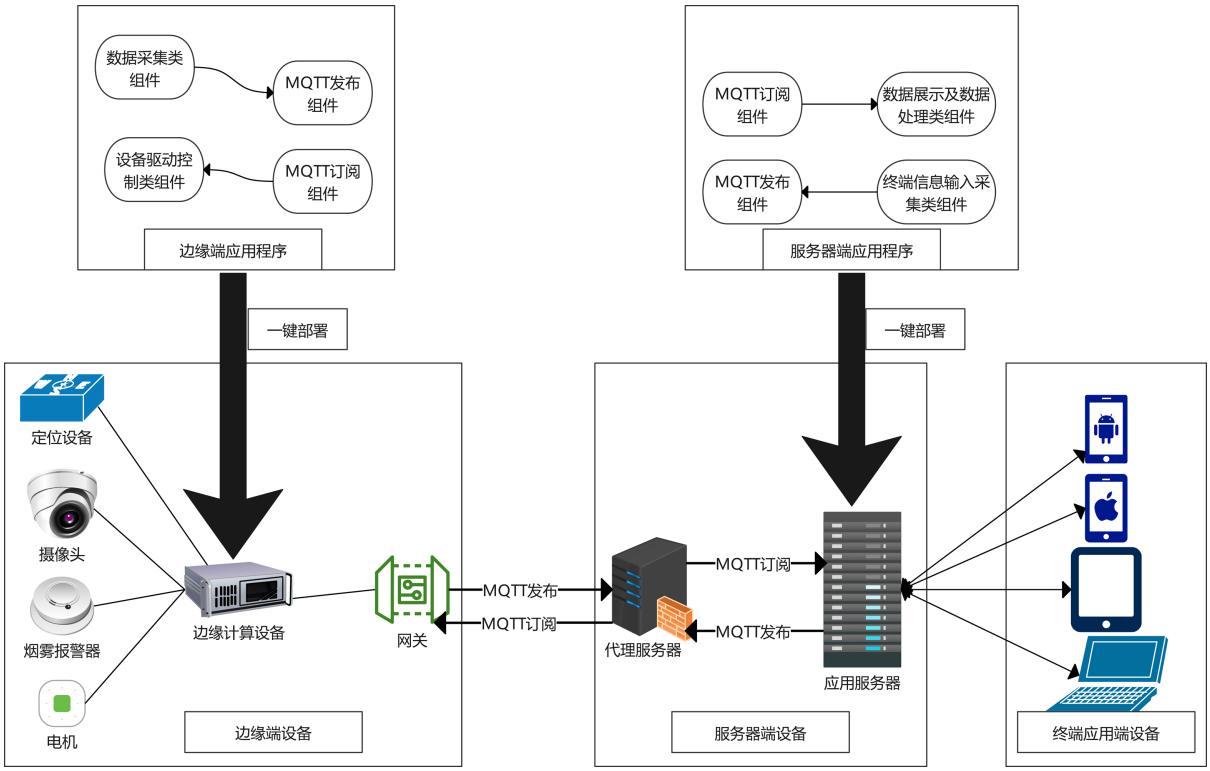

Figure 3. Component & M2M-based IoT application development methodology
图 3. 基于组件和 M2M 技术的物联网应用开发方法

边缘计算设备端运行的组件化软件应用程序，通过拖拽组件库中的组件可以快速完成软件的编写，快速高效，然后进行一键部署。代理服务器是一个 MQTT 代理服务器，可以将物联网系统中的边缘设备和应用服务器进行解耦，边缘端的组件软件应用程序的编写不需要考虑应用服务器的软件程序，应用服务器和边缘端计算设备的程序需要约定相应的主题。应用服务器端运行的组件化软件应用程序，通过拖拽组件库中的组件可以快速完成软件的编写，快速高效，然后进行一键部署。终端应用端设备通过 IP 和端口号即可访问到相应的物联网应用，不需要下载专用的 APP。这种基于组件和 M2M 技术的物联网应用开发方法可以实现边缘端和应用端的解耦，避免文本化开发和竖直型开发模式的不足。

基于组件化和 M2M 的物联网系统开发范式依赖于高效的数据交换和协作机制，以确保设备之间的信息交流和协同工作，该开发范式可用的数据交换和协作机制如下：

消息传递：M2M 通信使用消息传递作为主要的数据交换机制。通过定义规范的消息格式和协议，设备之间可以交换数据、状态信息和命令。消息传递机制支持异步通信和事件驱动的架构，使得设备能够以灵活的方式进行数据交换和协同操作。

发布/订阅模式：M2M 通信中常采用发布/订阅模式进行数据交换。设备可以发布数据或事件，其他订阅了相关数据或事件的设备可以接收并处理。这种模式支持实时数据更新和事件通知，使得物联网系统中的设备能够及时响应和协同工作。

远程过程调用：M2M 通信中的远程过程调用(Remote Procedure Call, RPC)机制允许设备之间通过网络调用和执行远程的方法或函数。这种机制使得设备可以远程控制其他设备的行为，实现分布式计算和协同工作。

数据同步和复制：M2M 通信中的数据同步和复制机制用于保持不同设备之间的数据一致性。当多个设备需要访问和操作相同的数据时，数据同步和复制机制可以确保数据的准确性和一致性，避免数据冲突和丢失。

协同决策和协作优化：基于 M2M 通信的物联网系统可以实现设备之间的协同决策和协作优化。设备可以共享数据和信息，通过分布式算法和协同优化技术进行联合决策和任务分配。这种协同机制可以提高系统的整体性能和效率，实现资源的最优利用。

6. 实验验证

6.1. 组件模型验证

为了验证本文提出的组件模型的可用性和运行效率，设计了一个累加器应用程序，可以实现从数字 1 累加到某个自然数，文本化开发方式实现的累加器作为对比。三个数据分别为：从 1 累加到 1000000 (不包括 100000)，从 1 累加到 100000000 (不包括 10000000)，从 1 累加到 10000000000 (不包括 1000000000)。对于从 1 累加到 1000000 (不包括 100000)，计算 10 次，运行时间求平均值，然后和文本化的累加器进行对比。对另外两个累加做同样的处理，对比试验结果见表 2。

Table 2. Component-based vs. Textual accumulators
表 2. 组件化和文本化的累加器对比

累加数值	文本化运行时间(毫秒)	组件化运行时间(毫秒)	文本化运行结果	文本化运行结果
Sum (1, 1000000)	5.725	5.957	500000500000	500000500000
Sum (1, 100000000)	130.439	137.016	5000000050000000	5000000050000000
Sum (1, 10000000000)	13597.595	14373.779	50000000005000000000	50000000005000000000

由计算结果和计算时间可知，两种方法的计算结果一致；对于计算时间，组件化累加器分别比文本化的累加器增加用时 4.367%、5.042%、5.708%。组件模型的有效性和可靠性得到了验证。

6.2. 基于组件和 M2M 技术的物联网系统构建方法验证

为了验证本文所提出的组件模型及物联网应用开发方法，搭建了如图 4 所示的组件化智能家居系统，

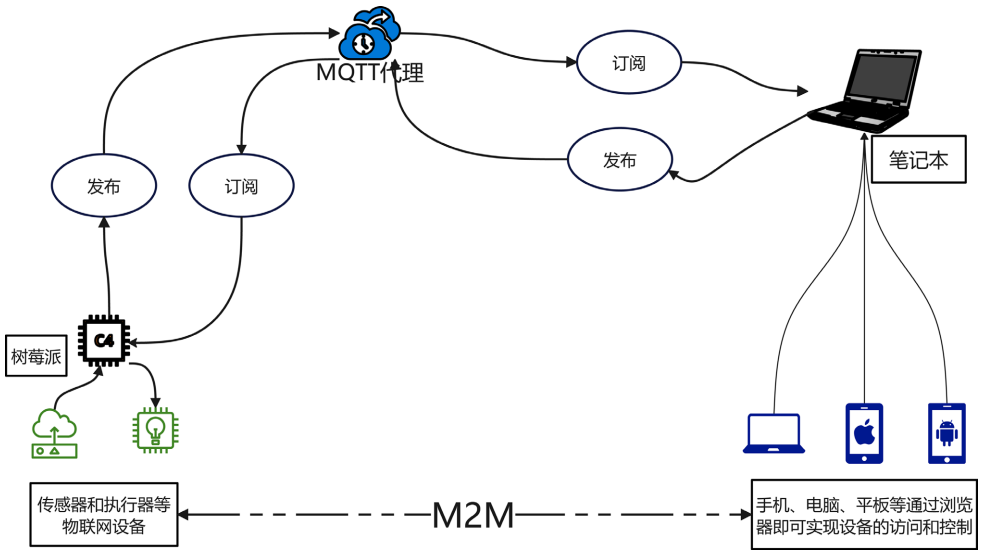

Figure 4. Architecture diagram of component-based smart home system
图 4. 组件化智能家居系统架构图

在系统中树莓派采集环境的数据，包括温度值、光照情况、以及是否有人按下门铃，然后进行发布。树莓派，根据订阅到的主题和根据环境对风扇、门铃和电灯做出相应的控制。代理服务器是 HiveMQ，一个企业级的 MQTT 代理。Windows 笔记本，作为应用服务器，根据订阅到的数据进行展示，同时还发布从终端应用端接受到的控制命令。手机、电脑、平板等通过浏览器即可实现设备的访问和控制，实现对设备的通信。系统中包括感知设备，执行设备，控制设备，还包括数据显示部分。本文的树莓派运行的系统是 RaspberryPiOS (64-bit)，基于 Linux 的操作系统，笔记本是基于 Windows10 的操作系统，手机是基于 Android 的操作系统，平板电脑是 IpadOS 16.5。图 5 和图 6 分别为组件化智能家居物联网系统的边缘端程序和服务器端程序。

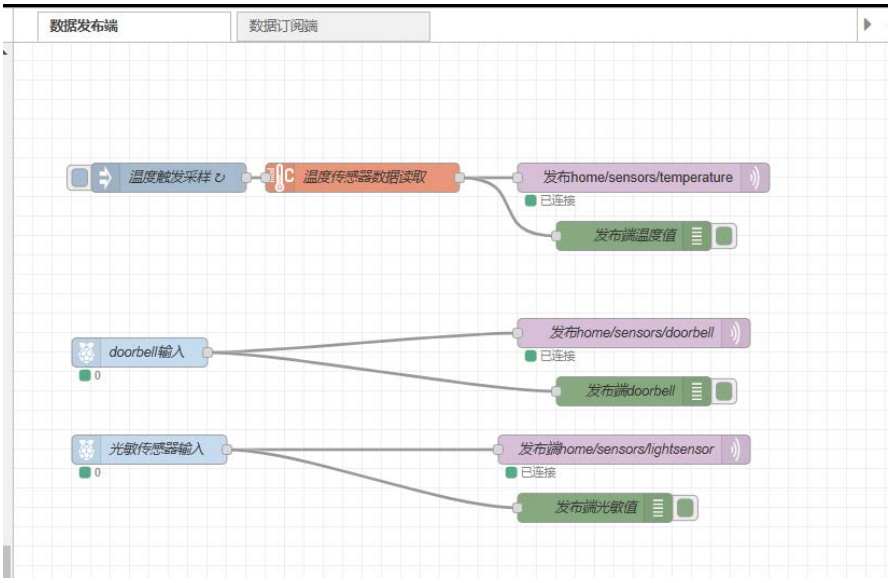

Figure 5. Edge publishing end for component-based smart home system
图 5. 组件化智能家居系统的边缘发布端

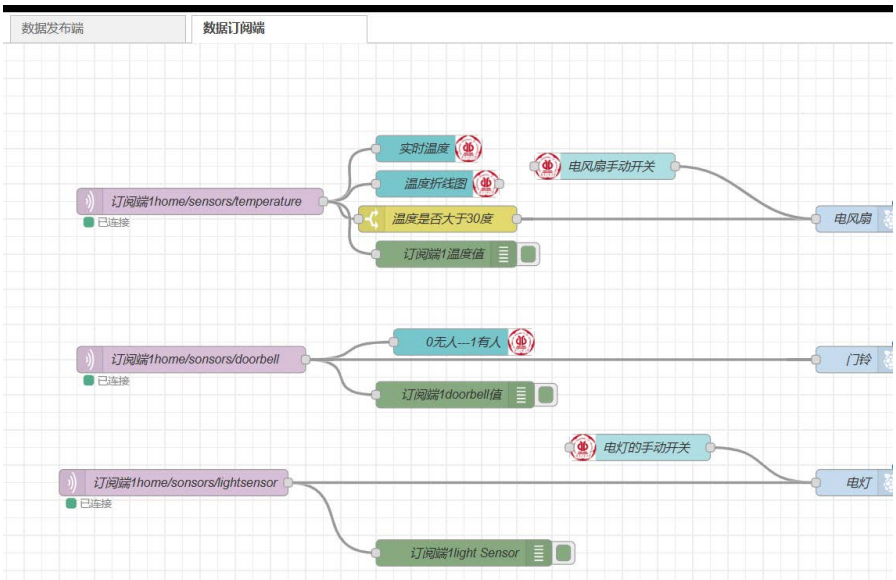

Figure 6. Application server side of component-based smart home system
图 6. 组件化智能家居系统的应用服务器端

在如图 7 展示的智能家居物联网系统的应用端界面，分为两个部分，数据显示部分和设备控制部分。对于数据显示部分，显示了温度曲线、门铃是否被按下、实时的温度数值。设备控制部分显示设备状态，还可以按下按钮来改变设备的状态。经过测试，数据显示部分和设备控制部分均有效。

Figure 7. Terminal application interface of component-based smart home system
图 7. 组件化智能家居系统的终端应用端界面

在组件化智能家居系统中，数据采集设备有三个：温度传感器 DS8B20 (连接树莓派的物理引脚 7)，触摸传感器 TouchSensor (代表门铃按钮，连接树莓派的物理引脚 37)，光敏传感器 LightSensor (连接树莓派的物理引脚 35)，树莓派将这些数据使用 MQTTout 组件进行发布，MQTT 代理是 HiveMQ，如图 8 所示的组件化智能家居系统的边缘节点的实物图。执行器包括蜂鸣器(用来代表门铃，连接树莓派的物理引脚 33)，红色的 LED (代表风扇，连接树莓派的物理引脚 29)，绿色的 LED (代表灯光，连接树莓派的物理引脚 33)。控制逻辑如下：

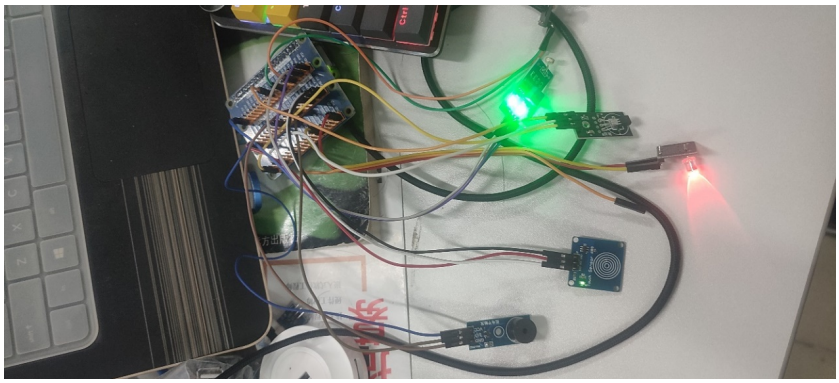

Figure 8. Physical picture of the edge end of a component-based smart home system
图 8. 组件化智能家居系统的边缘节点的实物图

如果传感器检测到温度大于 30 摄氏度，就可以自动地打开风扇进行降温。如果室内的人觉得太热需要降温，也可以通过浏览器访问设备的 IP 和端口号打开设备控制界面，手动的打开风扇进行降温。也可以手动的关闭风扇。

如果光敏传感器检测到天黑了，可以自动的开灯。同样的，也可以打开浏览器控制灯的开和管。

如果有人按下门铃按钮，蜂鸣器就会有响声。浏览器界面也会指出有人按了门铃。

7. 结论

在包含各种异构设备和数据的物联网应用开发中，M2M 通信将会在设备和设备交互、人机交互中扮演重要角色。本文提出的软件组件模型可以屏蔽系统开发过程中存在的平台、操作系统、编程语言、网络环境等高度异构的问题，实现应用程序的“Write Once, Run Anywhere”的能力。建立了组件化应用软件应用的模型，使用 DAG 图进行了形式化描述和验证，由本文提出的组件模型构成的应用程序符合有向无环图。提出了基于组件化的 M2M 的物联网系统开发范式，为高效快速的搭建跨平台、跨操作系统的物联网系统提供了可行的参考。开发累加器应用程序，对文本化编程和组件化编程进行比较，两种方法的计算结果一致；对于计算时间，组件化累加器分别比文本化的累加器增加用时 4.367%、5.042%、5.708%。组件模型的有效性和可靠性得到了验证。根据组件模型，开发了数据采集组件、数据处理组件、数据展示组件，这些组件是可以跨平台的，实现了“Write Once, Run Anywhere”的特点，搭建了组件化的智能家居系统，验证了组件模型的有效性，所搭建组件的可用性，在基于 Linux 系统的树莓派和 Windows 系统的笔记本上运行组件的物联网系统，验证了组件的跨平台性，以及验证了本文组件的 M2M 的物联网开发范式的可用性。

参考文献

[1] Tolba, A. and Al-Makhadmeh, Z. (2022) Modular Interactive Computation Scheme for the Internet of Things Assisted Robotic Services. *Swarm and Evolutionary Computation*, **70**, Article ID: 101043. https://doi.org/10.1016/j.swevo.2022.101043

[2] Choi, J., Park, J. and Pokhrel, S. (2021) Explore-before-Talk: Multichannel Selection Diversity for Uplink Transmissions in Machine-Type Communication. *Journal of Communications and Networks*, **23**, 23-33. https://doi.org/10.23919/JCN.2020.000029

[3] Chen, D.J. (2022) Towards a Software Component Framework for Mechatronics Applications. *Component-Based Software Engineering*. https://xueshu.baidu.com/usercenter/paper/show?paperid=4f61473a221e94deb4289084adb45eaa&site=xueshu_se&hitarticle=1

[4] Liu, C. (2021) Discovery and Quality Evaluation of Software Component Behavioral Models. *IEEE Transactions on Automation Science and Engineering*, **18**, 1538-1549. https://doi.org/10.1109/TASE.2020.3008897

[5] Göbel, R. and Kitzing, S. (2023) Defining Anonymity Properties of Data Sets with the Compliance Assertion Language (COMPASS). *Digital Government*: *Research and Practice*. https://doi.org/10.1145/3603255

[6] 童红兵. 组件技术发展与应用前景[J]. 宿州教育学院学报, 2012, 15(1): 137-140.

[7] 王欣. 基于 Web 服务和软件体系结构的构件组装技术的研究[D]: [硕士学位论文]. 哈尔滨: 哈尔滨工程大学, 2016.

[8] Samaddar, R., Ghosh, A., Sarkar, S.D., Das, M. and Chakrabarty, A. (2023) IoT & Cloud-Based Smart Attendance Management System Using RFID. *International Research Journal on Advanced Science Hub*, **5**, 111-118. https://doi.org/10.47392/irjash.2023.020

[9] Alturayeif, N., Alturaief, N. and Alhathloul, Z. (2020) DeepScratch: Scratch Programming Language Extension for Deep Learning Education. *International Journal of Advanced Computer Science and Applications*, **11**, 642-650. https://doi.org/10.14569/IJACSA.2020.0110777

[10] Marabesi, M. (2023) Towards a Web Architecture Structure Based on Intent. https://marabesi.com/pdfs/papers/PROWEB-2021.pdf

[11] Arbab-Zavar, B., Golestan, S., Vasquez, J.C. and Guerrero, J.M. (2023) Exploring Communication Architectures in Microgrids: Applications and Scenarios. *IEEE Industrial Electronics Magazine*. https://doi.org/10.1109/MIE.2023.3281717

[12] Reddy, Y.S., Dubey, A., Kumar, A. and Panigrahi, T. (2021) A Probabilistic Approach to Model SIC Based RACH Mechanism for Machine Type Communications in Cellular Networks. *IEEE Transactions on Vehicular Technology*, **70**, 1878-1893. https://doi.org/10.1109/TVT.2021.3055286

[13] Kościug, B., and Bilski, P. (2023) Energy Saving Chaotic Sequence Based Encryption, Authentication and Hashing for M2M Communication of IoT Devices. *International Journal of Electronics and Telecommunications*, **69**, 253-259.

[14] Bui, A.-T.H., Nguyen, C.T., Hayashi, T. and Pham, A.T. (2021) Small-Cell Assisted Group Paging for Massive MTC in LTE Networks: Design and Analysis. *IEEE Access*, **9**, 40933-40949. https://doi.org/10.1109/ACCESS.2021.3064321

[15] Prabhakara Rao, T. and Satyanarayana Murthy, B. (2023) Extended Group-Based Verification Approach for Secure M2M Communications. *International Journal of Information Technology*, **15**, 2479-2488. https://doi.org/10.1007/s41870-023-01284-w

[16] Xu, X.-R., Xu, Y.-H., Zhou, W. and Nallanathan, A. (2023) Energy Efficient Resource Allocation for UAV-Served Energy Harvesting-supported Cognitive Industrial M2M Networks. *IEEE Wireless Communications Letters*. https://doi.org/10.1109/LWC.2023.3278627

[17] Cao, N., Fineman, J.T., Russell, K. and Yang, E. (2022) I/O-Efficient Algorithms for Topological Sort and Related Problems. *ACM Transactions on Algorithms*, **18**, 1-24. https://doi.org/10.1145/3418356

[18] Aldana, J.A.A., Maag, S. and Zaidi, F. (2021) A Formal Consensus-Based Distributed Monitoring Approach for Mobile IoT Networks. *Internet of Things*, **13**, Article ID: 100352. https://doi.org/10.1016/j.iot.2020.100352