Embedded Technoloy and Intelligent Systems 嵌入式技术与智能系统, 2025, 2(2), 96-114
Published Online April 2025 in Hans. https://www.hanspub.org/journal/etis
https://doi.org/10.12677/etis.2025.22008

Hans汉斯

构建面向AGI时代的开源IoT AI智能体架构与实践

吴　薇[1,2]，曹宇伟[1]，区卉贤[2]，王坚豪[2]，徐朦饶[2]，胡雯轩[2]，陈睿轩[2]，邢成龙[2]，杨灵益[2]

[1]江苏矽望电子科技有限公司，江苏　南京
[2]杭州电子科技大学电子信息学院，浙江　杭州

收稿日期：2025年4月1日；录用日期：2025年4月23日；发布日期：2025年4月30日

摘　要

随着大语言模型(LLM)及边缘计算技术的发展，AI智能体(AI Agent)正逐步成为物联网(IoT)系统中的核心调度与控制单元。文章设计并实现了一套以AI智能体为核心的人工智能物联网(AIoT)系统架构，融合传感器/执行模块、边缘终端、本地/云端LLM推理引擎、云计算中心与n8n自动化平台。系统采用模块化设计，支持MCP调度架构与OPC UA、MQTT等协议通信，具备低延迟、高可扩展性和良好的工程可移植性。并重点介绍了系统构成、核心模块设计，以智能家具为典型应用实例及部署实验，展示其在各种AIoT行业应用场景下的实用性和开放性。

关键词

AI智能体，MCP，边缘计算，IoT模块，轻量级LLM，开源架构

Design and Implementation of an Open-Source IoT AI Agent Architecture for the AGI Era

Wei Wu[1,2], Yuwei Cao[1], Huixian Ou[2], Jianhao Wang[2], Mengrao Xu[2], Wenxuan Hu[2], Ruixuan Chen[2], Chenglong Xing[2], Lingyi Yang[2]

[1]Cynoware Electronics, Inc., Nanjing Jiangsu
[2]School of Electronic Engineering, Hangzhou Dianzi University, Hangzhou Zhejiang

Received: Apr. 1st, 2025; accepted: Apr. 23rd, 2025; published: Apr. 30th, 2025

Abstract

With the advancement of Large Language Models (LLMs) and edge computing technologies, Artificial

文章引用: 吴薇, 曹宇伟, 区卉贤, 王坚豪, 徐朦饶, 胡雯轩, 陈睿轩, 邢成龙, 杨灵益. 构建面向 AGI 时代的开源 IoT AI 智能体架构与实践[J]. 嵌入式技术与智能系统, 2025, 2(2): 96-114. DOI: 10.12677/etis.2025.22008

Intelligence Agents (AI Agents) are gradually becoming the core scheduling and control units in Internet of Things (IoT) systems. This paper designs and implements an AIoT system architecture centered around AI agents, integrating sensor/actuator modules, edge terminals, local/cloud-based LLM inference engines, cloud computing centers, and the n8n automation platform. The system adopts a modular design, supports the MCP scheduling framework, and communicates via protocols such as OPC UA and MQTT, offering low latency, high scalability, and strong engineering portability. This paper focuses on the system structure and core module design, and demonstrates its practicality and openness through a typical use case in smart home applications and deployment experiments, showcasing its potential across various AIoT industry scenarios.

Keywords

AI Agent, MCP, Edge Computing, IoT Module, Lightweight LLM, Open-Source Architecture

1. 引言

物联网(Internet of Things，文中简称 IoT)和人工智能(Artificial Intelligence，文中简称 AI)结合的人工智能物联网(文中简称 AIoT)作为当下智能社会的技术支柱，正与智慧城市与产业自动化等领域深度融合，广泛应用于工业制造、能源管理等对实时性与稳定性要求极高的行业，同时不断延伸至智能家居、适老健康监护等民生场景。然而，在生成式大语言模型 (Large Language Model，文中简称 LLM)取得突破之前，AIoT 系统受限于边缘设备的算力瓶颈与语义理解能力不足，主要依赖如 AlexNet、CNN 或客制化轻量化 AI 模型等进行感知与推理。这种技术路径导致 AIoT 系统普遍面临以下三大核心问题：一是任务固化(系统只能适应单一场景任务)；二是语义割裂(环境感知与决策过程相互分离，缺乏统一理解)；三是响应滞后(对用户动态需求和环境变化的适应能力有限)。

随着通用人工智能 AGI (Artificial General Intelligence，文中简称 AGI)技术的演进，AIoT 系统正迈向 AI 智能体(AI Agent，文中简称 AI 智能体或 AI Agent)发展，使得构建具备语义理解、工具组合调用与自我学习适应能力的“嵌入式智能体系统”成为可能。本文提出一种面向 AGI 时代的开源 AIoT 系统架构，集成 AI 智能体、MCP (Multi-Component Processing，多组件任务规划器，文中简称 MCP)中间件框架、IOT 边缘感知处理能力、轻量语言模型以及云平台可视化等功能，形成端－边－云协同的智能体系。本文以智能家居场景为例，对该系统进行了整个项目模块的集成实现与测试，验证了其在实际应用中的可行性与扩展潜力。项目软件系统将在 GITHUB 上公开，旨在与同行一起将此架构改进、完善并进行推广。作者单位之一的江苏矽望电子科技有限公司(文中简称矽望科技)将可为同行提供优惠的配套调试硬件。

2. IoT AI 智能体

IoT AI 智能体是一类部署于边缘或终端设备中的智能系统，具备环境感知、智能推理与行动执行三大核心能力。它融合了多模态感知技术与人工智能推理模型，能够对所处环境进行实时分析，并据此做出自主决策与响应操作，形成感知－决策－行动的闭环控制体系。

在功能上，IoT AI 智能体主要承担以下任务：

- 环境感知(Perception)：利用温度、湿度、图像、声音、光照等传感器采集环境信息，为后续推理提供

数据支持；

- 智能推理与决策(Reasoning and Decision-making)：通过部署轻量化 AI 模型(如卷积神经网络、小型语言模型或规则引擎)或通过本地 LLM 或云端 LLM 对感知数据进行分析，理解用户意图或判断环境状态，做出最优控制决策；
- 行动执行(Action Execution)：根据推理结果，驱动电机、继电器、显示器或控制信号，实现对外部世界的智能干预；
- 循环机制(Sense-Think-Act Loop)：上述功能以闭环形式持续运行，实现系统对动态环境的实时适应与响应。

该智能体系统的整体运行机制可概括为：“感知(Sense)→推理与决策(Think)→行动(Act)”，构成一个具有自适应能力的智能行为循环。其基本功能框图如图 1 所示，涵盖了从数据输入到控制输出的完整处理流程。

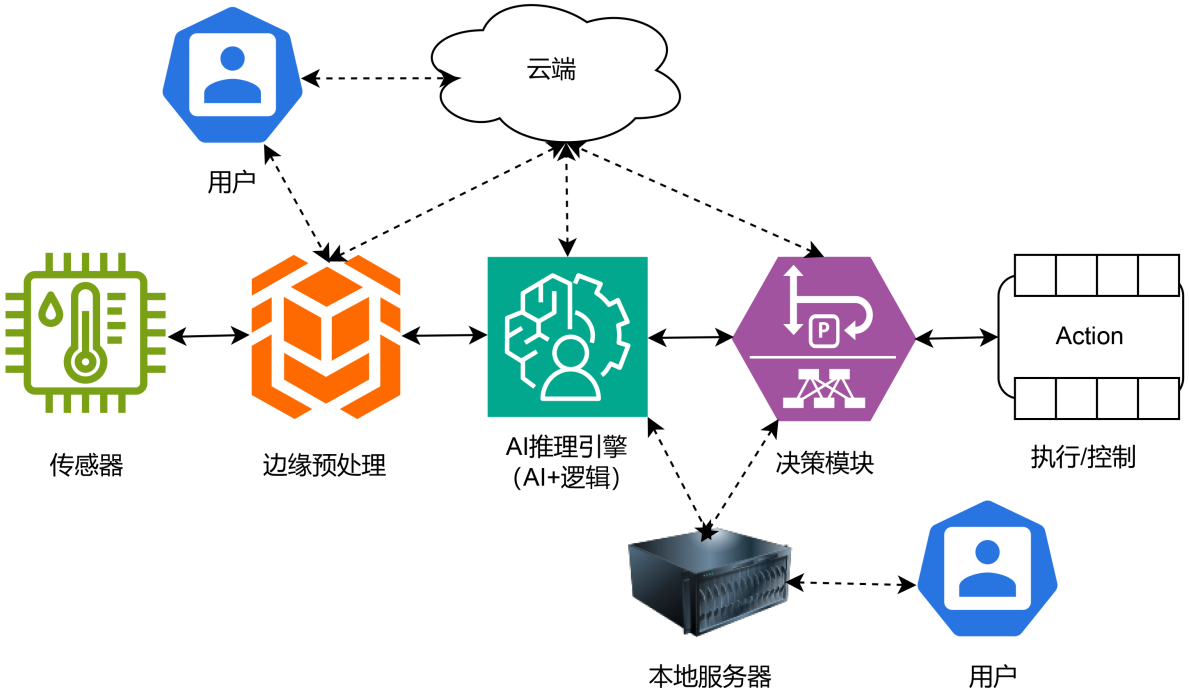

Figure 1. Functional block diagram of the IoT AI agent system
图 1. IoT AI 智能体系统的功能框图

通过引入 AI 智能体机制，IoT 系统的自主性与智能化水平得以显著提升，为智能家居、工业自动化、智慧医疗等领域带来了更具扩展性与适应性的解决方案。

3. 系统总体架构

典型的物联网架构有三、四、五、六、七层，其中图 2(a)的七层模型定义在论文[1]中有描述。值得指出的是，没有单一的、公认的物联网架构和层定义。根据特定的应用场景和业务任务的复杂性，物联网架构层数、层定义会有所不同[2]。

通过对图 2(a)进行简化，得到本文的 IoT AI 智能体系统架构，如图 2(b)所示，它包括传感器和执行模块、边缘终端、云平台、本地轻量级大语言模型 LLM、云端 LLM 和 AIoT 行业应用客户。

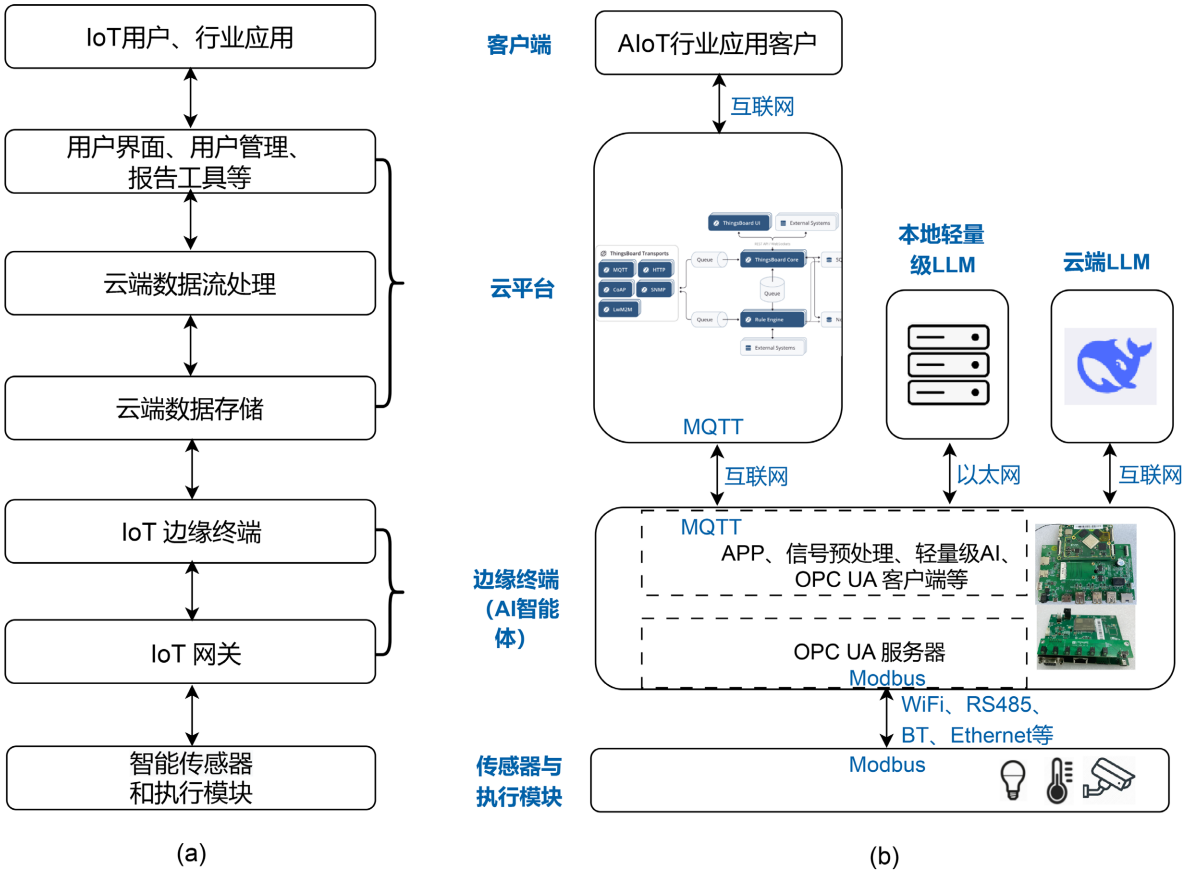

Figure 2. IoT AI agent system architecture
图 2. IoT AI 智能体系统架构

第一层传感器与执行模块和第二层具有物联网网关功能的边缘终端之间支持 Wi-Fi、蓝牙(BT)、RS485、以太网等多种无线和有线通信，建议主要使用 Modbus 数据通信协议。第二层和第三层云平台之间通信使用互联网的 MQTT 协议。第三层和第四层 AIoT 用户和行业应用客户端之间通信则可通过 WEB 客户端或 REST API 来实现。本地轻量级 LLM 可以使用 DeepSeek 小模型或 QwQ 等安装在本地服务器，以提高系统反应速度。表 1 描述了系统中各个组件的作用和示例。

Table 1. Components and functions of the IoT AI agent system
表 1. IoT AI 智能体系统的组件和作用

组件	作用	示例
传感器(输入)	从环境中收集数据	温度传感器、湿度传感器、运动检测器、摄像头等
AI 模型/算法(大脑)	分析数据并做出决策	机器学习模型、决策树、深度学习、大模型
本地处理(边缘计算)	在设备附近快速处理数据	单片机、Linux 终端、安卓终端等
执行器(输出)	根据决策执行动作	打开/关闭灯光、调节温控、发送通知等
通信模块	与其他设备或云端进行信息传输	Wi-Fi、蓝牙、Zigbee、MQTT、OPC UA、HTTP API、MCP 等
知识库(可选)	存储历史经验或预定义规则	本地数据库、云存储
用户界面(可选)	让用户可以与 AI Agent 交互	触摸屏、手机 App、语音指令等

4. 关键模块设计

4.1. 传感器和执行模块

第一层传感器和执行模块(文中简称 IoT 模块)主要完成对环境数据采集、信号预处理、执行器驱动及边缘终端通信[3]。如图 3 所示。该模块通过混合通信架构(无线/有线)与边缘计算终端进行数据交互，采用 Modbus 协议实现标准化数据传输，确保系统的兼容性与可扩展性。

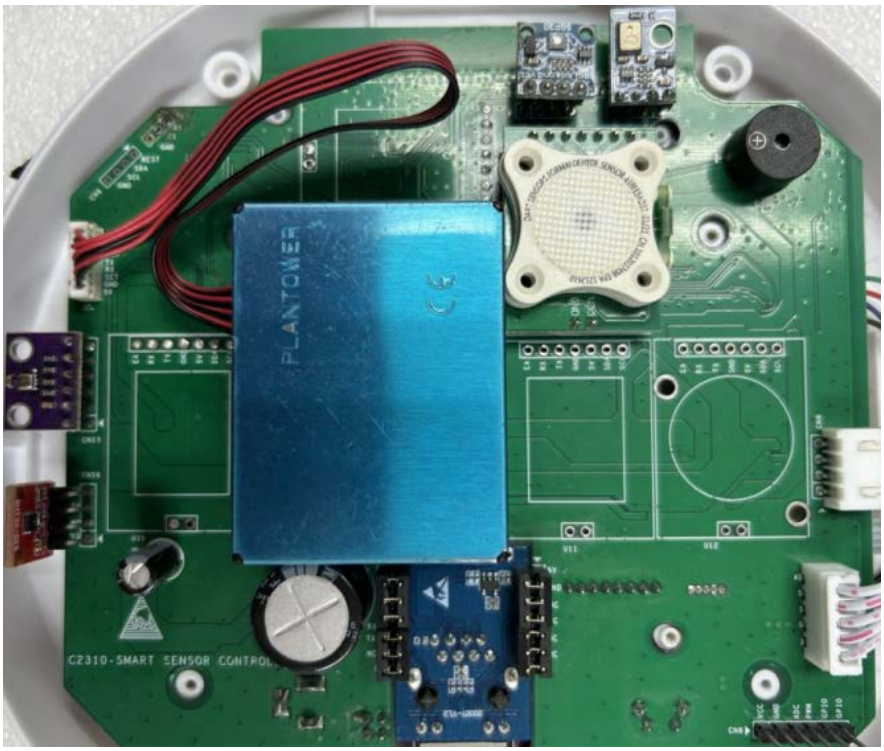

Figure 3. Sensors and actuator modules of the IoT AI agent system
图 3. IoT AI 智能体系统的传感器和执行模块

4.1.1. 硬件核心平台

IoT 模块采用国产 RISC-V 架构 W801 芯片作为主控单元，其技术特性包括：

- 处理性能：32 位 CPU 内核搭配 20KB ROM + 288KB SRAM + 2MB Flash 的三级存储架构，支持实时数据采集(采样率 ≥ 1 kHz)及简单算法如 PID 等控制算法的本地化执行。
- 接口扩展：集成 UART × 3、SPI × 2、I2C × 2、ADC × 5 等多类型数字接口，最大支持 10 个传感器/执行器的并行接入。
- 通信能力：双模无线支持(Wi-Fi 802.11b/g/n + BT/BLE 4.2)，有线接口包含 10/100M 以太网 PHY 和 RS485 收发器。
- 能效管理：动态电压调节技术实现待机功耗 < 10 μA，工作模式功耗 ≤ 120 mW@48 MHz。

4.1.2. 传感器与执行机构接口

模块通过标准化接口协议实现传感器网络的灵活配置：

- 模拟信号采集：16 位高精度 ADC 通道(采样率 1 MSPS)支持温度、湿度、光照等模拟传感器。
- 数字接口扩展：I2C 总线挂载大气压、CO_2浓度等数字传感器。

- SPI 接口连接工业级 IMU (惯性测量单元)。
- GPIO 驱动继电器、电磁阀等执行机构。
- 特殊信号处理：内置可编程增益放大器(PGA)支持 mV 级微弱信号检测。

4.1.3. 通信机制

模块采用分层通信架构：

- 物理层：无线传输选用 Wi-Fi (最大速率 72 Mbps)实现高带宽数据传输，BLE 4.2 用于低功耗设备组网，有线通信采用 RS485 (最大节点数 256 个，传输距离 1.2 km)。
- 协议层：基于 Modbus-TCP/RTU 协议实现数据帧标准化封装，定义设备状态字(16 bit)、传感器数据区(32 bit 浮点数组)、控制指令区(8 bit 命令码)三类寄存器。
- 安全机制：AES-128 硬件加密引擎保障数据传输安全，支持双向身份认证。

4.1.4. IoT 模块软件

W801 的 SDK 中内置有 FreeRTOS 实时操作系统。FreeRTOS 是一个开源嵌入式实时操作系统，其具有小巧简单、可移植性强的特点，使得它成为目前市场占有率最高的 RTOS (Real Time Operating System) [3]。FreeRTOS 支持多任务并发运行，它通过优先级管理、时间片轮转调度等机制管理多任务并确保 CPU 的高效利用。FreeRTOS 还提供了一系列的功能和服务，例如任务管理、时间管理、消息队列、信号量、互斥锁等，基于这些功能和服务开发人员能够更方便地进行嵌入式应用开发。

本文设计的 IoT 模块软件的任务流程如图 4 所示。软件启动后，系统创建了 N 个主要任务：任务 1 负责连接并监测网络，任务 2 负责监测 TCP 连接状态，任务 3 为 TCP 数据接收和处理，以及任务 N 等。

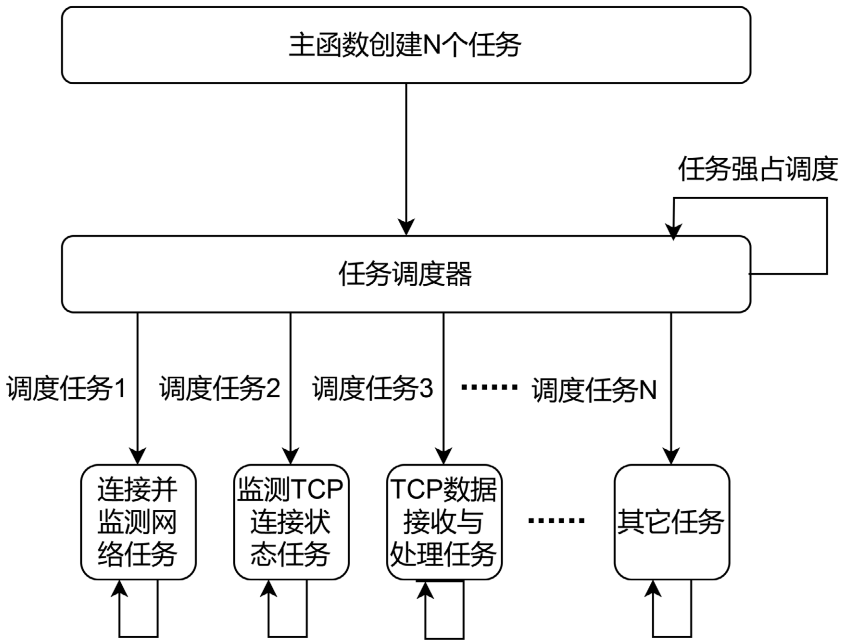

Figure 4. Task flow of the IoT module software
图 4. IoT 模块软件的任务流程

4.2. 具有物联网网关功能的 AI 边缘终端

边缘终端作为 AIoT 系统的核心边缘节点，不仅承担着多种通信协议(如 BT、Wi-Fi、RS485、Ethernet

等)下的设备接入与协同任务，还负责对环境数据预处理与协议统一。在 AI 能力层面，边缘终端集成本地轻量化 AI 模型(如 AlexNet、CNN 等)，用于进行初步的语义理解与智能推理，提升系统在断网或低延迟场景下的自适应能力。此外，本论文边缘终端内部通常还部署 MCP 与 AI 智能体执行框架，支持感知－推理－执行的闭环控制。其还内置 OPC UA 客户端/服务器、MQTT 客户端等标准工业协议模块，用于与 IoT 设备和云端平台(ThingsBoard)之间的数据交互与同步。

在面向 AGI 时代的系统演进中，边缘终端正从“数据中继器”转型为“本地智能体宿主”，具备更强的上下文理解、多模态感知与自主决策能力，成为真正意义上的“嵌入式认知节点”。

矽望科技近些年开发了采用十多款不同 CPU 的 Android 和 Linux 边缘终端。本文中系统由高性价比的国内主流终端芯片 RK3399 和 EC200A 的 Android 和 Linux 构成。

4.2.1. 边缘终端硬件

图 5 展示了矽望科技为支撑本文开源 IoT AI 智能体架构的边缘终端硬件系统框图，其核心以高性能嵌入式处理器(如 RK3399、EC200A 或其他 SoC)为中心，围绕处理器构建了五大模块：存储单元、网络与通信、用户接口、传感器扩展、电源管理。以下是各部分功能描述：

- 核心处理器(CPU)
  - 采用 Rockchip RK3399 或其他多核处理器，集成 GPU/NPU，支持本地 AI 推理。
  - 承担系统运行、数据处理、协议解析、AI 模型推理与边缘控制等核心任务。
- 存储模块(Storage)
  - DDR：用于系统运行所需的临时内存。
  - iNand：嵌入式非易失性存储，存放操作系统、模型与应用程序。
  - TF Card：支持外置存储卡扩展，便于数据日志存储与软件更新。
- IoT 网关与互联网通信(IoT Gateway & Internet)
  - RS485/RS232：适配传统工业串口设备，支持 Modbus 等协议。
  - Wi-Fi & BT：实现无线连接与蓝牙设备通信。
  - Ethernet：用于有线高速网络连接。
  - USB：支持外设接入(如摄像头、传感器、调制解调器等)。
  - 3G/4G/NB-IoT：提供广域网远程通信能力。
  - RFID/NFC：支持本地身份识别、门禁或物品追踪等应用。
- 用户交互接口(User Interfaces)
  - LCD + 触摸屏：实现可视化图形界面与人机交互。
  - Audio + Audio Amplifier + Microphone + Earphone：构成音频输入输出系统，支持语音识别与播报。
  - Camera：用于视觉识别、人脸检测等。
  - HDMI：可外接大屏幕显示。
  - Keys：支持按键操作输入。
- 电源管理与扩展接口(PMIC, Power, Extension)
  - PMIC(电源管理 IC)：用于稳定供电、充电管理与电池保护。
  - Power & Battery：支持外部直流供电与电池供电两种方式，满足移动或断电场景需求。
  - Extension：保留扩展接口，用于功能增强或定制化开发。

该边缘终端具备多协议兼容、多模态感知、本地 AI 推理与人机交互能力，可广泛应用于智能家居、工业自动化、智慧医疗等 AIoT 场景。它是实现“感知－推理－执行”智能体闭环的关键硬件基础。

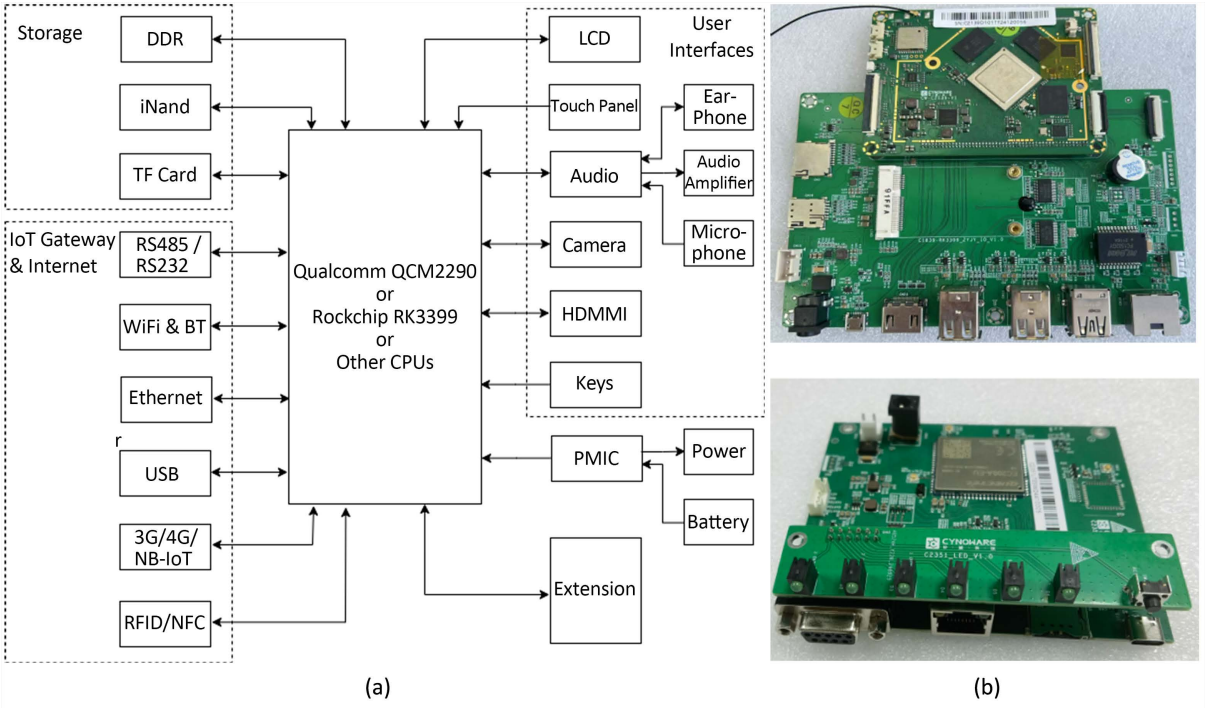

Figure 5. AIoT edge device hardware system
图 5. AIoT 边缘终端硬件系统

表 2 是基于 RK3399CPU 的边缘终端硬件规格书。

Table 2. Hardware specifications of the edge device based on RK3399 CPU
表 2. 基于 RK3399CPU 的边缘终端硬件规格书

Items	Product	Cynoware C2139 Edge Device
Main	CPU	RK3399 Dual-core Cortex-A72 + Quad-core Cortex-A53
	RAM	4 GB RAM
	Storage	16 GB
	LCD	10.1" (1920 × 1200) option
	Touch	10.1" multi-touch TP option
	Camera	1080p option
Audio	Speaker	2 W × 2 option
	Mic	×1 option
Connectivity	WIFI	IEEE802.11 b/g/n (2.4G) and BT 4.2
	Ethernet	1 × RJ45 with 10/100/1000 M
IO	USB	Type A USB 3.0 × 1, USB 2.0 × 1, Micro USB 2.0 × 1
	DC In	Input: AC 100~240 V; Output: DC 24 V/3.75 A
OS	OS	Android 10

4.2.2. 边缘终端软件系统架构

边缘终端不仅作为物理交互节点和 IoT 数据汇聚平台，其软件系统也承担着 AI 推理、协议桥接、任务调度与用户交互等关键职责。整体软件架构由操作系统、通信中间件、智能体执行模块、用户接口与

自动化工作流组件构成，主要包括以下部分：

- 操作系统层

边缘终端运行 Android 或嵌入式 Linux 操作系统，提供图形界面支持、硬件抽象层与网络协议栈，支撑设备驱动与 AI 模型的运行环境，图 6 所示的是 AIoT Android 系统。

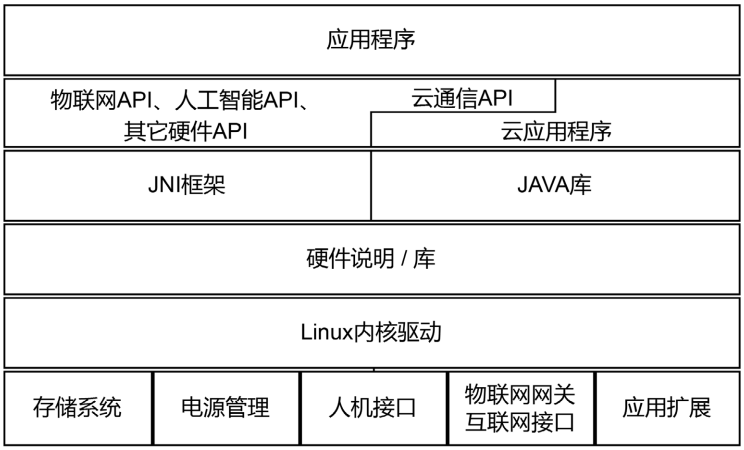

Figure 6. Android operating system of the AIoT edge device
图 6. AIoT 边缘终端的 Android 操作系统

- 通信中间件
  - MQTT 客户端：用于边缘终端与本地服务器(如 ThingsBoard)及云端平台的数据发布与订阅，实现轻量级的物联网通信。
  - OPC UA 服务器&客户端：支持工业标准信息建模与设备抽象，提供传感器/执行器的统一接口管理与上下行通信能力，可适配 Zigbee、RS485、Modbus 等底层协议。
- AI 智能体模块
  - 集成本地 AI 模型(如 AlexNet、MobileNet 或自定义轻量化卷积神经网络)，用于场景感知、行为识别、环境预测控制等任务。
  - 具备初级类人对话能力与自主任务规划能力，支持实时数据分析与事件驱动行为决策。
  - 当本地 AI 算力无法满足需求时，可调用远程 LLM (如 DeepSeek 等)完成复杂推理或任务分解，体现出智能体的任务迁移与资源感知能力。
- 用户交互界面
  - 提供基于语音识别与语音合成的自然语言交互入口，用户可通过口头指令控制设备或查询状态。
  - 集成触摸屏 GUI 与可视化仪表盘，展示环境数据(如温湿度、光照、空调状态等)，并提供手动控制选项。
- 自动化与工作流引擎(n8n)
  - 内嵌或集成开源自动化平台 n8n，用于构建事件驱动型业务流程，实现任务的条件判断、工具链调用与动作序列自动执行。
  - 支持自定义插件与 Webhook 接口，可与 AI 智能体模块协同触发复杂自动化响应。
- 任务调度与边缘－本地服务器协同机制
  - 边缘终端内置任务调度器，根据任务类型与资源占用情况，判断是否本地执行或委托本地服务器/云端推理处理。

- 实现“感知–判断–执行”闭环智能体循环的同时，具备跨层协同能力，适应多任务并发与动态场景变化。

4.3. 云平台模块：ThingsBoard [4]云端服务器

在整个系统中，云平台承担着数据汇聚、远程控制、可视化展示与智能分析等关键职能。本系统基于开源物联网平台 ThingsBoard 搭建云端服务器，结合 Dash 平台实现高级数据分析和多维度监控。

- 通信机制
  - 通过 MQTT 协议与边缘终端实现双向通信，支持设备数据上报与命令下发，确保系统低延迟、轻量化、稳定的网络交互。
- 核心功能(基于 ThingsBoard)
  - 设备管理与配置：支持批量设备注册、分组、远程参数配置与生命周期管理。
  - 实时数据监控：采集温度、湿度、光照、人体感应等 IoT 数据，支持图表化、仪表盘展示。
  - 数据存储与历史记录：可自定义数据保留周期与存储格式，支持 SQL/NoSQL 数据库集成。
  - 规则链引擎与自动化控制：内置流式处理机制，支持条件判断、联动控制、告警触发等自动化逻辑。
  - 远程控制与命令执行：用户可通过 Web 端或移动端下发开关、调节等控制指令至边缘设备。
  - 权限与安全管理：内置角色分级、Token 认证、访问日志记录，确保数据隐私与设备安全。
  - 性能分析与可扩展性：支持多租户部署，便于在工业、智能家居、智慧城市等场景下横向拓展。
- Dash 高级可视化平台
  - 通过集成 Dash (Plotly)或 ThingsBoard 内置高级可视化插件，展示 AI 推理结果、设备状态、历史趋势与预测数据，支持用户自定义仪表盘与动态图表组件。
- 远程访问与用户交互
  - 用户可通过 ThingsBoard Web 仪表盘或移动端应用，进行远程设备管理、参数查询、告警响应与命令交互，提升系统整体的可操作性与可维护性。

4.4. 本地与云端 LLM 协同推理模块

在面向 AGI 的 AIoT 系统中，大语言模型(LLM)作为核心的认知与推理引擎，为 AI 智能体提供语义理解、任务规划、自然交互等高阶能力。考虑到不同应用场景对延迟、算力和网络的要求差异，本系统设计支持本地 LLM 与云端 LLM 的协同部署机制，由 MCP 动态调度调用。

- 本地 LLM (轻量化 LLM 如 QwQ)
  - 部署位置：位于边缘设备或局域网内服务器。
  - 模型类型：通常为小参数量模型(如 TinyGPT、MiniLM、DistilBERT 等)，支持基础问答、指令解析与简易对话。
  - 优势：
    - ✓ 响应速度快，推理延迟低(毫秒级)；
    - ✓ 数据本地处理，保障隐私；
    - ✓ 可在网络不可用或不稳定时持续运行；
    - ✓ 适用场景：设备控制、单轮指令理解、环境反馈解析等对实时性要求高的任务。
- 云端 LLM (如 DeepSeek 等)
  - 部署方式：通过 API 接口远程访问云计算平台上的大型预训练语言模型。
  - 优势：
    - ✓ 模型规模大，具备更强的语言理解、推理和多轮对话能力；

- ✓ 支持知识检索、复杂逻辑推理、多任务规划等高级功能；
- ➢ 劣势：
  - ◆ 网络依赖性高，存在一定时延；
  - ◆ 数据需传输至云端，可能存在隐私风险；
  - ◆ 适用场景：复杂场景感知、跨模态分析、长期任务管理、人机自然语言交互等。

- MCP 调度策略
  - ➢ AI 智能体内嵌 MCP (多组件规划器)，可根据任务特征动态选择推理引擎，策略包括：
    - ■ 任务复杂度：简单任务由本地 LLM 处理，复杂推理任务委托云端 LLM；
    - ■ 实时性要求：对延迟敏感的任务优先采用本地推理；
    - ■ 网络状态感知：在弱网或离线情况下自动回落至本地模型；
    - ■ 模型可用性：基于设备资源(内存、CPU 占用等)判断本地模型是否可运行。

本文充分结合边缘计算与云计算优势，实现高性能－低延迟－高智能的 AIoT 认知系统，为 AI Agent 在 AGI 时代的泛化部署奠定基础。

5. MCP、AI 智能体和 n8n

在面向 AGI (通用人工智能)时代的 AIoT 系统中，实现系统级智能不仅依赖于单点设备的局部推理能力，更需要一种能够协调多源感知、模型资源、执行工具的智能体协作机制。为此，AI 智能体和 MCP (Multi-Component Planner)作为 AIoT 智能体架构中的两个核心角色，分别承担“智能终端交互”与“系统任务规划与调度”的职责。AI Agent 聚焦于环境感知与用户交互，具备初步的认知能力与本地决策功能；而 MCP 则作为智能中枢，负责对高层语义任务进行结构化分解、模型选择与跨模块执行调度。

5.1. 定义

1) AI Agent [5] (人工智能智能体)

AIoT 系统中的 AI Agent 是部署在边缘设备或云端的智能软件体，具备环境感知、语义理解、自主决策与动作执行能力，是 IoT 系统中“智能行为的实施者”。它通常集成轻量级模型或本地推理引擎，负责接收来自用户或环境的输入信息，并根据内置策略或外部规划结果完成特定任务。AI Agent 表现出一定程度的类人智能，能够持续与用户交互、自我适应，并保持上下文理解。

2) MCP [6] [7] (Multi-Component Planner，多组件任务规划器)

MCP 是部署在边缘终端或局域网服务器上的任务调度与智能规划核心模块，主要负责将 AI Agent 所提出的高层语义任务转化为可执行的多步骤任务流程，并动态调用工具、模型或服务来完成具体动作。MCP 实现了“感知－规划－推理－执行”智能体循环机制中的“推理－规划”中心，是整个系统的智能控制中枢。

3) n8n [8]

n8n (Node-Node)是一个开源、低代码的工作流自动化平台，允许用户通过图形化界面将不同的应用服务、API 接口与逻辑操作串联起来，构建自动化任务流程。n8n 支持超过 300 种第三方集成，具备高度可扩展性与自托管能力，广泛应用于数据同步、自动控制、智能通知、系统协调等场景。

AIoT 中的 MCP 体系通过将任务逻辑抽象成可规划的组件，解耦了智能体的感知与执行能力，以及任务推理和资源协调能力，实现了“轻智能终端 + 重调度中心”的系统模式。MCP 服务器与客户端之间的高效协同，是构建支持 AIoT 智能体系统的基础。而在 AIoT 系统中，n8n 被用作任务执行与工具编排层，配合 AI Agent 与 MCP 结构，完成传感器数据处理、控制指令下发、模型调用与结果反馈等功能，成为智能体系统中“执行”的核心组件之一。

5.2. AIoT MCP

之所以称为 AIoT MCP 是因为本文 MCP 用于 AIoT 系统中。本文 AIoT MCP 的架构设计中采用“服务器–客户端(Server-Client)”结构，主要是为了实现分布式协同、模块解耦、可扩展性强以及异构设备之间的统一通信与管理。

MCP 系统采用集中式调度 + 分布式协作的体系结构，由部署在边缘终端或局域网本地服务器的 MCP 服务器负责全局任务协调与资源分配，部署在各边缘终端的 MCP 客户端实现感知上报、任务请求与结果接收，本文 AIoT MCP 的架构如图 7 所示。

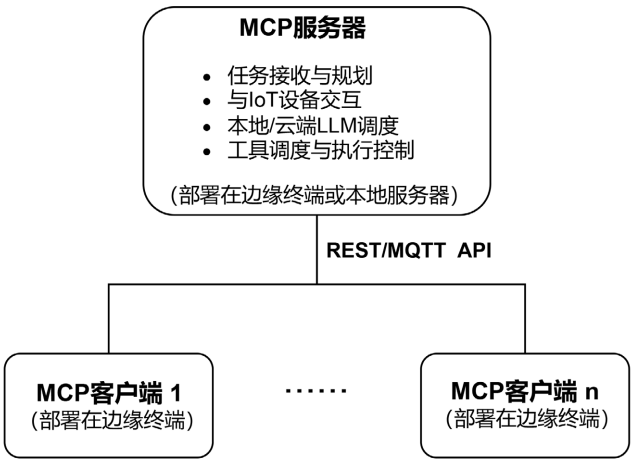

Figure 7. AIoT MCP architecture
图 7. AIoT MCP 架构

1) AIoT MCP 服务器(MCP Server)

- 核心功能
  - 任务接收与解析：接受 AI Agent 或应用层提交的自然语言或结构化任务请求。
  - 意图识别与任务规划：通过内置解析器与规划器(支持 HTN/PDDL/LLM 生成计划)将任务拆解为步骤序列。
  - 模型选择与调用：根据任务复杂度、网络状态及资源占用动态选择本地 LLM 或云端 LLM 进行推理。
  - 工具调度与控制：通过注册表统一管理所有工具模块，并通过 OPC UA、MQTT、REST 等协议执行任务。
  - 日志与状态反馈：记录任务执行过程与结果，支持返回至客户端或上传至云端平台(如 ThingsBoard)。
- 技术架构
  - 构建语言：FastAPI/Flask
  - 调度策略：基于规则 + LLM 增强计划(如 Plan-GPT)
  - 模型对接：本地 Llama.cpp/云端 DeepSeek/OpenAI API
  - 工具接口：JSON-RPC/OPC UA SDK/MQTT broker
  - 数据缓存：Redis + SQLite

2) AIoT MCP 客户端(MCP Client)

- 部署位置
  - 部署在**每个边缘终端设备(如智能屏、嵌入式网关)**上，与 AI Agent 运行在同一节点。
- 核心功能
  - 感知信息上传：从传感器采集数据(如温湿度、占用状态)并打包上传至 MCP 服务器；
  - 任务请求发起：当 AI Agent 识别到用户意图或环境事件后，构造任务请求并发送给 MCP 服务器；
  - 响应接收与执行：接收 MCP 返回的执行计划或最终动作命令，并调用本地工具或控制 API 执行；
  - 执行反馈回传：将执行状态或结果回传给 MCP 服务器，或通过 MQTT 同步至云平台。
- 示例通信流程

3) 协同机制与部署策略

MCP 服务器和客户端的协调机制和部署策略如表 3 所示。

Table 3. Coordination mechanism and deployment strategy of the MCP server and client
表 3. MCP 服务器和客户端的协调机制和部署策略

项目	MCP 服务器	MCP 客户端
推荐部署位置	边缘服务器或本地主控节点	每个 AI Agent 边缘终端
可服务对象	多个 MCP 客户端/Agent	对接单一 MCP 服务器
通信协议	REST API, MQTT, WebSocket	REST API/MQTT
安全机制	Token 认证 + IP 白名单	内置 Agent 身份标识
容器化建议	支持 Docker Compose 部署	作为边缘服务模块启动

5.3. AIoT AI 智能体

1) AIoT AI 智能体定义

之所以称为 AIoT AI 智能体是因为本文 AI 智能体是部署于 AIoT 系统边缘终端中的自治智能体模块，具备以下核心能力：

- 环境感知：从传感器收集温湿度、光照、人体存在等多模态数据；
- 语义理解：识别用户语音或文本输入的意图；
- 本地推理：基于轻量模型(如 TinyML、MiniLM、DistilBERT)进行本地判断；
- 用户交互：通过语音、图形界面或触摸交互与用户对话；
- 行动触发：根据理解结果或来自 MCP 的任务响应，执行动作控制或信息反馈。

AI Agent 是系统中任务的感知入口与行为执行者，它既能独立执行简单任务，也能将复杂任务委托给 MCP 系统。

2) AI 智能体功能结构

AI 智能体(AI Agent)功能结构如表 4 所示。

Table 4. Functional structure of the AI agent
表 4. AI 智能体功能结构

AI 智能体(边缘端)
• 感知模块(传感器数据采集)
• 语义识别模块
• 本地推理模块(轻量模型)
• UI 交互模块(触摸/语音界面)
• MCP 客户端接口(请求和回传)

3) AI Agent 与 MCP 的协同关系

AI Agent 并不直接承担复杂的任务规划与多模块调度，而是与 MCP 系统协同工作，形成以下表 5 所示的分工。

Table 5. The collaborative relationship between AI Agent and MCP
表 5. AI Agent 与 MCP 的协同关系

功能阶段	AI 智能体	MCP 客户端	MCP 服务器
感知与输入	采集环境数据、接收用户请求	封装任务请求并上传	接收请求、解析任务、生成计划
意图分析	初步识别指令或语义内容		高阶意图识别/任务分解
任务请求	调用本地 MCP 客户端	向 MCP 服务器发起任务请求	根据上下文选择工具/模型资源
推理/规划	若为简单任务则本地完成	负责传输任务结果或指令	复杂推理由本地/云端 LLM 完成
执行与反馈	控制设备或提示用户	上报执行状态	日志记录与反馈回传至智能体

AI Agent 是边缘侧的智能“执行终端”，MCP 是系统的智能“调度中枢”。通过将任务规划与推理能力上移至 MCP，Agent 设计得以更轻量化、响应更实时；通过将感知与执行下沉至 Agent，系统整体智能具备更强的实时性、鲁棒性与可扩展性。两者的分工与协作，共同构建了适应 AGI 时代的分布式 AIoT 智能体系统。图 8 所示的是 AI 智能体与 MCP 的协同机制。

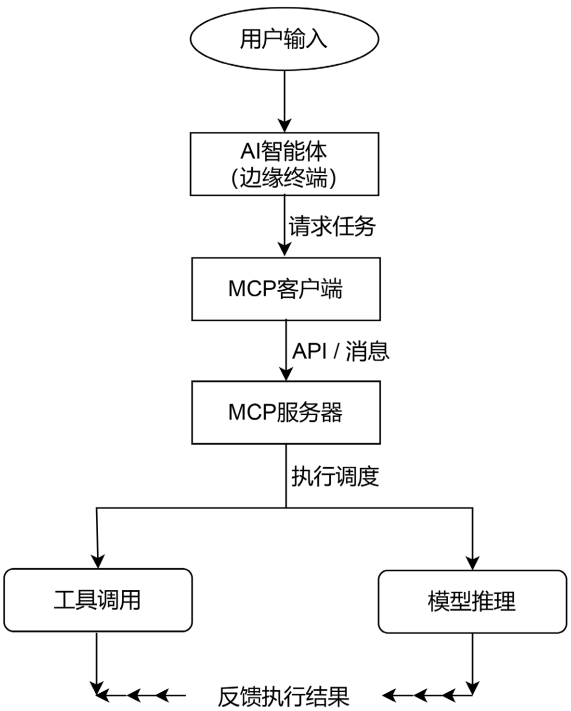

Figure 8. Collaboration mechanism between the AI agent and MCP
图 8. AI 智能体与 MCP 的协同机制

5.4. n8n 的主要功能

1) n8n 是一款开源的低代码自动化工作流引擎，其主要功能如表 6 所示。

Table 6. Main functions of n8n
表 6. n8n 主要功能

功能模块	功能描述
可视化任务流程设计	通过图形界面拖拽节点，构建自动化流程，无需编码
多协议接口集成	原生支持 HTTP、MQTT、WebSocket、数据库、文件系统等接口
Webhook 监听与触发	可作为外部系统任务入口点，接收来自 AI Agent 或 MCP 的请求
逻辑判断与条件控制	支持 if/switch/merge 等分支逻辑，完成任务路径动态选择
定时与事件驱动执行	可根据时间、事件或传感器状态自动执行任务流
插件与扩展机制	支持自定义 Function 节点(JS 脚本)，扩展逻辑处理能力

2) n8n 的作用

在本 AIoT 系统架构中，n8n 的核心作用可概括为以下几点：

- 任务执行器：根据 MCP Server 下发的计划任务，逐步执行各类工具控制、环境数据获取与状态反馈；
- 工具调用调度层：封装调用设备控制 API、模型推理服务、外部天气服务等操作，使任务执行与 AI Agent/MCP 解耦；
- 条件驱动的自动化引擎：根据实时环境或外部信息动态决定执行路径，如：白天拉开窗帘，夜间调暗灯光；
- 任务反馈通道：将任务执行结果上传至云平台(如 ThingsBoard)或返回给 AI Agent，完成“行动–感知–反馈”闭环。

3) 与 AI Agent 和 MCP 的协同关系

n8n 与 AI Agent 和 MCP 构成了 AIoT 系统中的感知–推理–执行三层协同结构，其具体关系如表 7 所示。

Table 7. The collaborative relationship between n8n, AI Agent, and MCP
表 7. n8n 与 AI Agent 和 MCP 的协同关系

层级	模块	作用
感知/交互层	AI Agent	接收用户输入(语音/图形界面)、获取环境数据、提出任务意图
推理/调度层	MCP Server	分析任务目标，调用本地/云端模型进行推理与任务规划，生成执行计划
执行/反馈层	n8n	根据计划调用设备控制工具与 API，实现动作执行与状态同步

6. AI 智能体与 MCP 驱动的个性化客房舒适控制：智慧酒店应用实例

6.1. 应用背景

在智慧酒店场景中，不同客人入住不同房型时，对环境舒适度的需求具有较强的个性化。系统需根据用户偏好、客房实时环境数据、酒店地理位置及室外气候进行综合判断，实现如“调整房间舒适度”这类模糊语义指令的智能响应。

6.2. 场景示例

- 酒店地点：杭州(CN-Hangzhou)
- 当前时间：夏季，室外温度 34℃，湿度 78%

- 客房号：Room 102
- 用户：外籍商务客人，偏好干爽、静音环境

6.3. 用户请求

用户通过床头屏/语音助手输入指令：
“请把房间调得舒服一点。”

6.4. 系统模块组成

酒店实例的系统模块组成如表 8 所示。

Table 8. System module composition of the hotel use case
表 8. 酒店实例的系统模块组成

模块	功能说明
AI 智能体	接收用户指令，提取意图，调用 MCP 客户端
MCP 客户端	打包任务，上传至 MCP 服务器
MCP 服务器	任务规划，调用外部气候信息，生成执行计划
n8n	工作流引擎，编排工具调用、条件判断、执行流程自动化
IoT 执行设备	空调、除湿机、窗帘、灯光，通过 MQTT/OPC UA 控制

6.5. n8n 实现流程(结构图)

酒店实例的 n8n 流程如图 9 所示。

Figure 9. n8n workflow of the hotel use case
图 9. 酒店实例的 n8n 流程

6.6. 实例任务数据传递结构

来自 AI Agent 的请求(Webhook 输入)：

```
{
  "task": "adjust_room_comfort",
  "room_id": "102",
  "location": "CN-Guangzhou",
  "user_pref": {
    "temperature": 25,
    "humidity": "dry",
    "light": "soft"
  }
}
```

6.7. 外部气候信息查询节点

GET https://api.weatherapi.com/v1/current.json?key=API_KEY&q=CN-Hangzhou

返回示例：

```
{
  "location": { "name": "Hangzhou" },
  "current": {
    "temp_c": 34.2,
    "humidity": 78,
    "is_day": 0
  }
}
```

6.8. 生成执行计划(Function 节点逻辑)

```
const weather = items[0].json.current;
const plan = [];
if (weather.temp_c > 28) {
  plan.push({ device: "ac", action: "set_temperature", value: 25 });
}
if (weather.humidity > 65) {
  plan.push({ device: "dehumidifier", action: "on" });
}
if (!weather.is_day) {
  plan.push({ device: "curtains", action: "close" });
  plan.push({ device: "light", action: "dim", value: 30 });
}
return plan.map(p => ({ json: p }));
```

6.9. 控制指令节点(HTTP/MQTT)

示例：

POST http://iot-gateway.local/device/control

```
{
  "device": "ac",
  "command": "set_temperature",
  "value": 25,
  "room_id": "102"
}
```

6.10. Agent 回传反馈内容节点

```
{
  "message": "房间温度已设为 25℃，除湿设备已开启，灯光已调暗。祝您入住愉快！"
}
```

6.11. 智能特性总结

酒店实例的智能特性如表 9 所示。

Table 9. Intelligent features of the hotel use case
表 9. 酒店实例的智能特性

智能特性	实现机制
个性化调节	用户偏好 + 当前房间数据
场景感知(夜晚)	天气 API 中的 is_day
环境适应	实时气候推理 + 动作规划
多工具协同控制	n8n 流程中串行 + 条件控制
云边协同	本地 Agent 交互 + 云端 MCP 知识调用

该实例展示了在智慧酒店环境中，AI Agent 与 MCP 的协同配合如何通过 n8n 实现面向 AGI 风格的“任务理解–情境推理–自动执行”闭环流程。系统不仅能响应自然语言的模糊请求，还能动态适应城市气候、夜晚时段等条件，体现了 AIoT 智能体系统的高度扩展性与泛化能力。

7. 总结及展望

7.1. 总结

本文围绕面向 AGI (通用人工智能)时代的 IoT 智能体系统，提出并实现了一种开源、可扩展、模块化的 AIoT 系统架构。该架构以 AI Agent 与 MCP (多组件任务规划器)协同机制为核心，融合边缘计算、本地/云端推理、自动化流程引擎(n8n)与 IoT 控制协议(如 MQTT、OPC UA)等关键技术，具备以下显著优势：

- 感知–推理–执行闭环：实现从用户意图输入到物理动作输出的完整智能体工作流；
- 任务解耦与模块协同：AI Agent 负责交互与触发，MCP 负责任务规划与工具调度，二者职责明确、解耦部署；

- 多模态融合与动态推理：系统可集成环境感知、位置与气候数据，动态选择本地/云端模型，提升系统智能响应能力；
- 开源与低耦合：各模块接口统一，基于开源技术构建，便于复用与工程扩展；
- 应用落地性强：通过智慧家庭与智慧酒店实例，验证架构在真实场景下的实用性与可部署性。

通过上述研究与实践，本文验证了构建面向 AGI 能力的 AIoT 系统在当前技术条件下的可行路径，为实现具备泛化理解与复杂任务自主执行能力的嵌入式人工智能系统奠定了基础。

7.2. 展望

尽管本系统已在架构与功能上初步实现 AGI 智能体的核心特征，但仍存在值得进一步研究与优化的方向：

- 本地 LLM 模型的轻量化与定制化。面向边缘设备 LLM (如 TinyGPT、Phi-2)的部署仍面临资源瓶颈，后续需探索蒸馏、量化、微调等模型优化技术以提升推理性能与本地自主能力。
- 任务规划器智能化升级。当前任务规划基于规则与轻量 LLM 混合方式，未来可引入更强的推理规划模型(如 Plan-GPT、AutoGPT)，增强系统在多轮任务理解与未知任务应对上的泛化能力。
- 多 Agent 协同与分布式智能体调度。当多个 AI Agent 部署在同一建筑或系统中，如何实现分布式 MCP 协同与任务抢占、冲突解决将成为重要的研究方向。
- 安全与隐私保护机制。在多 Agent 与云平台协同过程中，需进一步加强身份认证、数据加密与边缘隐私计算，确保系统运行的可靠性与合规性。
- 行业级部署与开源推广。后续将计划以 Docker + n8n + LLM 为基础封装系统的核心能力，发布开源框架、服务智慧园区、养老照护、智能制造等 AIoT 垂直领域。

参考文献

[1] 吴薇, 吕亚欣. 服务国外市场的 AIoT 开源方案及其应用[J]. 单片机与嵌入式系统应用, 2022, 22(9): 4-9.

[2] Santos, M.G.D., Ameyed, D., Petrillo, F., Jaafar, F. and Cheriet, M. (2020) Internet of Things Architectures: A Comparative Study. arXiv:2004.12936.

[3] 琚子晗, 白贺, 杨喜童. 基于 Freertos 与 ARM 的智能探索机器人系统设计与实现[J]. 机械工程师, 2021(6): 37-39+42.

[4] ThingsBoard Inc. (2023) Things Board Open-Source IoT Platform Documentation. https://thingsboard.io/docs/

[5] Masterman, T., Besen, S., Sawtell, M. and Chao, A. (2024) The Landscape of Emerging AI Agent Architectures for Reasoning, Planning, and Tool Calling: A Survey. arXiv:2404.11584.

[6] Krishnan, N. (2025) Advancing Multi-Agent Systems Through Model Context Protocol: Architecture, Implementation, and Applications. arXiv:2504.21030.

[7] Hou, X., Zhao, Y., Wang, S. and Wang, H. (2025) Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions. arXiv:2503.23278.

[8] n8n GmbH (2024) n8n Workflow Automation Documentation. https://docs.n8n.io